AutoCAD

שימושים
מתקדמים



AUTOCAV אותו־קו

הנדסה ממוחשבת בע"מ

AutoCAD

סדנא לשימושים מתקדמים

אותו קו הנדסה ממוחשבת בע"מ

מרכז ההדרכה

קרליבך 27, ת"א טל. 5615156

1991

ה ת ו כ ן

עמוד

AUTOLISP.

<u>**נושאים נכללים :**</u>

קשר בין AUTOCAD ל- AUTOLISP.
הגדרת בעיה ותכנון תוכנית.
תיעוד ואחזקה.
תהליך ה- EVAUATION ב- ALISP.
מבני אינפורמציה ב- ALISP.
פעולות אריטמתיות.
השמת משתנים.
הגדרת פונקציות.
משפטי תנאי.
לולאות.
גישה לישויות בדיקה ושינוי.
גישה לטבלאות .
קלט פלט ב- ALISP.
תוכניות דוגמא ב-AUTOLISP.

תומרה

CPU -

יחידת עיבוד מרכזית - מבצעת את פקודות המכונה. מעבדים מסווגים על פי מהירות העיבוד ועל פי מספר ה-BIT ב-BUS שלהם.
משפחות מעבדים מבית INTEL - 80X87 80X86 8088.

זכרון RAM -

זכרון המחשב לקריאה וכתיבה, נמדד באלפי בתים BYTES (KB).
סטנדרט ל-DOS ל- 640 KB, ניתן להרחבה עד ל-16 MB. המידע נשמר עליו כל זמן שהמחשב פועל.

זכרון ROM -

זכרון קריאה בלבד, צרוב ברכיבים אלקטרוניים. כולל בו את התוכנה הטוענת את מערכת ההפעלה.

דיסקים וטייפים -

התקנים אלקטרו מכניים, המבצעים שמירת מידע גם כאשר המחשב אינו פועל. דיסקים מסווגים על פי כמות אינפורמציה ומהירות גישה. קיימת הפרדה בין דיסק קשיח לבין דיסקט. הדיסקטים, בנוסף להיותם ניידים הינם פחות אמינים בתור אמצעי איכסון, ובתור שכאלה - מומלץ לגבות אותם בכמה העתקים. טייפים מצטיינים באמינות שמירת מידע מול זמן גישה ממושך.
אמצעי חדש הינו COMPACT DISK, בעל יכולת לשמירת כמויות עצומות של מידע.

מדפסות -

מתחלקות על פי שיטת הדפסה: LASER ,DOT MATRIX, INK JET.

עכבר ומספרת (דיגטייזר) -

מאפשרים הצבעה ל-AUTOCAD. קיימים שני סוגי עכברים - אופטי ומכני.

תווינים -

מתחלקים לקבוצת תוויני עטים ותווינים אלקטרוסטטיים.
מתאמי מסך.

תוכנה

מערכת הפעלה - MS-DOS

מערכת ההפעלה מקשרת בין החומרה לבין תוכנות המשתמש. מערכת ההפעלה מספקת שרותים לתוכנות השונות. בנוסף, היא מבצעת קיטלוג קבצים, עריכת DIRECTORY, בדיקת דיסקים ועוד.

תהליך BOOT -

כאשר אנו מדליקים מחשב, או לחילופין מקישים על כפתורי CNTL-ALT-DEL, אנו מבצעים תהליך המכונה BOOT. המחשב טוען תוכנית זעירה, המצויה בתוך ה-ROM שלו אל ה-RAM ומתחיל בביצועה : תחילה מתבצעת סידרת בדיקות, כמו: בדיקת זכרון, בדיקת הקונפיגורציה מול הקונפיגורציה השמורה ב-ROM, בדיקת אמצעי אחסון. לאחר מכן, נטען ה-DOS מתוך הדיסק הקשיח, נטען קובץ CONFIG.SYS, ובסיום מופעלות פקודות קובץ AUTOEXEC.BAT.

פקודות DOS נפוצות:

- COPY

העתקת קובץ, אופצית VERIFY.

- RESTORE ו- BACKUP

העתקת קבצים רבים, עם אפשרות להעתיק קבצים גדולים מדיסקט עד DOS 3.2. ניתן לראות את שמות הקבצים המגובים.

- TIME DATE

פקודות לקביעת תאריך ושעה במחשב - חשוב לצרכי מעקב אחר גירסאות של שרטוטים.

- CHKDSK

פקודה לבדיקת מצב קבצים שאינם ניתנים לגישה בדיסק. הפקודה מאפשרת "ניקוי" איזורים בדיסק. אין להפעיל אותה מתוך SHELL של ה- ACAD או של תוכנה אחרת.

- XCOPY

מאפשר העתקה מהירה של קבצים ו-DIRECTORY שלמים.

- PATH

הפקודה מציינת ל-DOS היכן ובאיזה סדר לחפש קבצים, שאינם נמצאים ב-DIR העבודה הנוכחי.

- PROMPT

הפקודה קובעת מה ייכתב בשורת ה-DOS על המסך. הכיתוב הסטנדרטי הינו <שם הדיסק:\שם ה-DIR> --- C:\WORK>. אך ניתן להוסיף תאריך,שעה שם מחשב, או קונפיגורציה נוכחית של המחשב.

- CD MD RD

אוסף פקודות ליצירה ביטול ומעבר לתוך DIRECTORY.

- PRINT

פקודת הדפסה המעבירה את הפיקוד של ההדפסה ל-DOS כאשר אפשר להמשיך בעבודה .

<u>סביבת העבודה של ה-AUTOCAD</u>

זכרון ה -ACAD -

נזקק לצורך עבודה רגילה ל- 512K זכרון. לעבודה עם AUTOLISP ל- 640K.
התוכנה תשתמש בתוספת זכרון להקטנת מספר הגישות לדיסק וקיצור זמן תגובה.
ה-ACAD יכול להשתמש ב - 4 MB נוספים של זכרון EXTENDED או EXPANDED.
כאשר ה-AUTOLISP EXTENDED יכול להשתמש ב- 14 MB זכרון נוסף.

זכרון סטנדרטי -

זכרון רגיל הינו עד 1 MB, והוא ניתן לגישה ישירה במחשבי AT. 640 KB ראשונים הינם לשימוש המערכת ותוכנות משתמשים, ו- 384 KB עבור תוכנות מערכת, כמו: תוכנות ROM או DRIVER של כרטיסי מסך.

זכרון EXTENDED -

זכרון המצוי בין 1 MB ל-16 MB וניתן לגישה בשיטת PROTECTED MODE של AT בלבד. רוב מחשבי AT יכולים להשתמש בזכרון זה ע"י שמירת נתונים ב-RAM DISK. ACAD יכול להשתמש בזכרון זה.

זכרון EXPANDED -

מאפשר לתוכנות מסוימות שנכתבו על פי תקן LIM להשתמש ב-16 MB זכרון. מערכות אלה מופעלות על ידי תוכנות מערכת מיוחדות. ACAD יכול להשתמש בזכרון מסוג זה.
לצורך שיפור תגובת ה-ACAD, יש להגדיר את ניצול הזכרון בצורה נכונה.
ניתן לחסוך זמן יקר בעבודת ה- ACAD כאשר מנצלים את הזכרון ביעילות.

CONFIG.SYS -

הינו קובץ בו אנו מציינים ל-DOS פרמטרים עבור סביבת העבודה שלנו. בזמן BOOT, ה-DOS מחפש את הקובץ ב-ROOT DIRECTORY. כמה מהפקודות בו יכולות להאיץ את זמן תגובת המחשב. כדי להפעיל שינויים במערכת, יש לבצע BOOT לאחר שינוי קובץ ה- CONFIG.SYS.

פקודות בקובץ CONFIG.SYS:

BUFFERS=מספר

FILES=מספר

- BUFFERS

פקודת BUFFERS=20 גורמת ל-DOS להקצות 20 שטחים של 512 BYTES (כלומר 10K BYTES) עבור קריאה וכתיבה לדיסק. השטחים בזכרון מנוצלים בצורה שלא כל פעולת קריאה וכתיבה מצריכה גישה פיזית לדיסק, פעולה המאיטה את מהירות התגובה של המערכת. מצד שני, ריבוי BUFFERS תופס מקום בזכרון . מומלץ ערך בין 10 לבין 35. לבדיקה, מומלץ לטעון קובץ גדול (מעל 256K בגודל), ולמדוד את זמני הטעינה וה-REGEN עבור ערכים שונים של BUFFERS.

- FILES

פקודת FILES=20 - FILES לא תופס כמעט מקום בזכרון. הפקודה מאפשרת טיפול ב-20 קבצים בבת אחת. זאת מאחר ו-DOS מגביל את מספר הקבצים היכולים להיות פתוחים ברגע נתון.

- VDISK

במקרה שיש זכרון EXTENDED, ניתן להגדיר את כולו או חלקו בתור דיסק נוסף.
ACAD בודק זכרון זה ומשתמש אוטומטית בחלק שאינו מנוצל.
בכדי להגדיר RAM DISK, יש לציין ב-CONFIG.SYS שורה (בתלות בגרסת DOS) DEVICE=VDISK.SYS 2500 /E פקודה המגדירה דיסק :E בגודל 2.5 MB.

<u>ניהול הזכרון ע"י ACAD</u>

ACAD משתמש ב- 4 MB זכרון מקסימום, EXTENDED או EXPANED, עבור I/O PAGE SPACE. כאשר עורכים שרטוט קטן הוא נשמר כולו בזכרון. ככל שגדל השרטוט ונעשה מורכב, נפתחים קבצים זמניים, וחלקים של השרטוט נכתבים לדיסק כדי לפנות מקום בזכרון לחלקים אחרים. תהליך זה מכונה PAGING. כדי להקטין למינימום את מספר הגישות לדיסק ולהביא לתגובה אופטימלית, מומלץ להקצות כמה שיותר זכרון עבור I/O PAGE SPACE.
ACAD בודק אוטומטית קיום זכרון נוסף וכן את סוגו, ומשתמש ב- EXPANDED (במידה ומותקנים שניהם) בעדיפות ראשונה.

<u>קובץ AUTOEXEC.BAT</u>

כאשר נמצא הקובץ ב-ROOT DIRECTORY, הוא מבוצע באופן אוטומטי בעת ביצוע BOOT. בקובץ מקובל להכניס פקודות המגדירות סביבת עבודה (כמו פקודות SET,PATH,PROMPT, ולהפעלת התמיכה העברית במחשב), וכן לטעינת תוכניות TSR מסוג DRIVERS לעכבר ומסך, או תכניות שירות מסוג SPOOLER.

קובץ AUTOEXEC.BAT אופייני ייראה כך:

PATH C:\DOS;C:\ACAD;C:\;C:\UTIL	
EGAHE	טעינת עברית
PROMPT SYSTEM1..$P..$G	הודעת המערכת ברמת DOS
SET ACAD=D:\ACAD	ציון מיקום תוכנת ACAD
SET ACADCFG=C:\ACAD\CFG	ציון מיקום קונפיגורצית ACAD
SET ACADFREERAM=20	שטחי עבודה בזיכרון
SET LISPHEAP=35000	
SET LISPSTACK=10000	
CD \WORK	

<u>פרמטרים המוגדרים עבור ה-ACAD:</u>

- ACAD

היכן מותקנים קבצי ה- ACAD אם אינם ב- DIR הנוכחי.

- ACADCFG

מאפשר לעבוד עם מספר תצורות של התוכנה, בלי לבצע שינוי תצורה אלא פשוט להצביע אל קובץ CFG אחר.

.SET ACADCFG=\MATROX

- ACADFREERAM

קובע את גודל הזכרון הנשמר ע"י ACAD כשטח עבודה.שטח זה מדווח ע"י פקודת STATUS כ-FREE RAM. ברירת המחדל 24K אך ניתן לקובעו יותר קטן או יותר גדול. כאשר מגדילים את ה- FREE RAM יש הגדלה בשטח העבודה אך לעומת זאת שטחי זכרון עבור PAGING קטנים. דבר זה הגורם לביצוע פעולות קריאה/כתיבה לדיסק. לרוב, אין סיבה להגדיל את ה- ACADFREERAM משום שהתועלת בכך קטנה. במקרים בודדים מומלץ להגדיל את ה- ACADFREERAM, והם: שימוש רב ב-SPLINE FITTING עם מספר רב של SPLINESEGS, (מספר הSEGMENTS שיווצרו עבור כל SPLINE PATCH) משום ש- ACAD משתמש באלגוריתם מהיר יותר של חישוב SPLINE, אם יש לו מספיק שטח עבודה בזכרון.פקודות נוספות שאינן מתפקדות היטב תחת שטח זכרון חסר הן: HIDE TRIM ו-OFFSET.

הגדלת שטח ה-PAGING (בהקטנת ACADFREERAM מתחת לברירת המחדל) משפרת את תפקוד ה-ACAD. בעזרת פקודת DOS : SET ACADFREERAM=nn כאשר גודל השינוי לא יעבור את ה-10K BYTES. רק בניסוי ניתן למצוא את הכמות האופטימלית של זיכרון, הדרושה לתצורת העבודה של כל משתמש.

למשל כתיבת SET ACADFREERAM=28 ברמת DOS או ב- AUTOEXEC.BAT תגרום ל-ACAD לשמור 28K BYTES עבור שטחי עבודה. שמירת שטח קטן מידי תגרום ב-ACAD להודעה FATAL ERROR: OUT OF RAM ויציאה ל-DOS . במקרה זה יש להגדיל את ערך ה-ACADFREERAM ולנסות שוב להעלות את ה-ACAD. במיקרים נדירים נרצה לרדת מתחת לברירת המחדל של 24, או לעלות על ערך של 30. בכל מקרה שנגדיר ערך גדול מהאפשרי בתצורת המחשב שלנו, יגיב ה-ACAD כאילו הגדרנו את המקסימום האפשרי.

- ACADXMEM

מאפשר לקבוע במדויק באיזה שטחים בזכרון ה-EXTENDED ישתמש ה-ACAD עבור I\O PAGING. אם לא מצוין ACADXMEM משתמש ה-ACAD בכל כמות הזכרון האפשרית. ציון הפרמטר מאפשר שימוש בתוכנות אחרות, המשתמשות בזכרון EXTENDED באותו זמן .

```
SET ACADXMEM=start,size

SET ACADXMEM=166K,256K
```

- ACADLIMEM

הגדרה דומה עבור זכרון EXPANDED.

- LISPSTACK-ו LISPHEAP

AUTOLISP הינה שפת תכנות הכלולה ב- ACAD. קיימת גירסת EXTENDED AUTOLISP היכולה להשתמש בזכרון EXTENDED עבור **LISPHEAP**. זאת למרות ש-640K מספיקים לרוב השימושים. משתני ופונקציות AUTOLISP נשמרים בזכרון למשך זמן הפעולה של ה-ACAD.
אם אין די זכרון לטעינת ה-LISP, יש הודעה MEMORY-AUTOLISP DISABLED .INSUFFICIENT

בהפעלת AUTOLISP, הוא מקצה שני שטחי זכרון: **HEAP** ו- **STACK**.

- HEAP

בו נשמרות פונקציות ומשתנים (NODES).

- STACK

בו נשמרים ארגומנטים של פונקציות ותוצאות חלקיות.
ביצוע יותר NESTING (סוגריים בתוך סוגריים) - דורש יותר זכרון ל- **STACK**.

ברירות מחדל

HEAP = 40000 BYTES

STACK = 5000 BYTES

סה"כ מוקצים בזכרון מקסימום 45000 BYTES לשניהם.

LISPXMEM- מאפשר להגדיר במדויק היכן יטען ה-EXTENDED AUTOLISP יחד עם שטחי ה-**HEAP** וה-**STACK**.
בכך ניתן לארגן את שטחי ה-EXTENDED גם עבור ה-ACAD וגם עבור תוכנות אחרות הפעילות באותו זמן.

<u>סכימת מבנה הזיכרון</u>

כתובת 0 בזכרון.---------------------------------

DOS,CONFIG.SYS, DEVICE DRIVERS

TSR & ADI DRIVERS

תוכנת

שרטוט - AUTOCAD

שטחי עבודה FREERAM (64K)

תוכנה AUTOLISP

HEAP 45K

יחד STACK - (64K)

AUTOCAD I/O PAGE

עד לגבול 640K.

כתובת 640K בזכרון ----------------------------------

שמור עבור DOS

כתובת 1MB --

EXTENDED\EXPANDED

RAM DISK (VDISK)

LOTUS,DB3

AUTOCAD EXTENDED I\O PAGING SPACE

עד 16MB ---

תוכניות EDITOR,

<u>מטלה</u>: תרגול שימוש ב-EDITOR

כתוב קובץ בשם TEST.EX1,הכנס לתוכו את התוכן הבא:

FIRST LINE

SECOND LINE

TIHRD LINE

FOURTH LINE

שמור את הקובץ, הדפס אותו למסך בעזרת פקודת **TYPE**.

הכנס חזרה ל-**TEXT EDITOR** ושנה:

1. לאחר השורה השלישית הכנס שורה: THIS LINE IS AFTER THREE.
2. בשורה FOURTH LINE,שנה הכתוב ל-NEW FOURTH LINE.
3. העתק שורה ראשונה לסוף הקובץ.
4. העתק את הקובץ כולו לסופו (שיכפול).
5. מחק את שורה 2 עד 4 ושמור הקובץ.
6. הדפס על המסך בעזרת TYPE.

"גזירת ותפירת" תוכנת ACAD.

הודעת הפתיחה ב-ACAD - קובץ **ACAD.MSG** כולל את הודעת הפתיחה בכניסה לתוכנה הוא ניתן לשינוי או ביטול .

מטלה : בעזרת ה-**EDITOR**, החלף את ההודעה הסטנדרטית להודעה:

WELCOME TO OUR ADVANCED SEMMINAR.

הכנס ל-AUTOCAD ובדוק את התוצאה.

הגדרת קוים LINETYPES

תוכנת ה-ACAD כוללת ספריית קווים סטנדרטים,עליה ניתן לשנות ולהוסיף. הגדרות הקווים נמצאות בקובץ ACAD.LIN.

כל הגדרה מורכבת משתי שורות :

שורה ראשונה : * - לציון הגדרה ושם הקו אחרי פסיק תיאור לצרכי תיעוד ללא פונקציה.

שורה שניה : A ואחריו ציון אורכי הקטעים (בספרות)-
מספר חיובי - קו
מספר שלילי - רווח
אפס - נקודה

המספרים כולם יחסיים, והם נקבעים על פי הפרמטר של ה-ACAD **LTSCALE** בזמן השרטוט.
ההגדרה דורשת שסימון יתחיל בקו או נקודה .

```
*DASHED,__ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

A,.5,-.25

*HIDDEN,_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

A,.25,-.125

*CENTER,____ _ ____ _ ____ _ ____ _ ____ _ ____ _ ____

A,1.25,-.25,.25,-.25

*PHANTOM,_____ _ _ _____ _ _ _____ _ _ _____ _ _ ____

A,1.25,-.25,.25,-.25,.25,-.25

*DOT,.........................................

A,0,-.25
```

<u>שיטה אחת ליצירת קו חדש</u>

ברמת ACAD נכתוב:

```
COMMAND  :  LINETYPE
?/CREATE/LOAD/SET   :   C
```

ACAD דורש שם, **סוג קו ושם קובץ**, על מנת לשמור את ההגדרה.

<u>שיטה אחרת</u> : כניסה בעזרת EDITOR לקובץ ACAD.LIN והוספת קו בו.

<u>מטלה:</u> הגדר קו חדש בשם NEWLINE בצורה ___ ... _ . ___ ... _ .
אחסן אותו בקובץ והעלה אותו לשרטוט קו.

<u>הגדרת HATCH PATTERNS</u>

ה-ACAD כולל ספרית HATCH PATTERNS סטנדרטיים ואפשרות לשנות, להוסיף ולגרוע ממנה. הם נמצאים בקובץ ACAD.PAT, מוצג כאן חלק ממנו.

כל הגדרה (בדומה להגדרת הקווים), מתחילה ב-*, שם ה-**HATCH** והערה מתארת.

השורות הבאות בהגדרה בנויות במבנה הבא:

ANGLE,X-ORIGIN,Y-ORIGIN,DELTAX,DELTAY,DASH1,DASH2

ראשית, יש להבין כי מדובר בהגדרת מספר קוים נפרדים ושונים, המצטרפים לצורה אחת. כל שורה בהגדרה מתארת קו אחד. נכללות התכונות הבאות: זוית הקו, ראשית הקו ב-X וב-Y, קידום יחסית לאותה נקודת ראשית ותיאור הקו על פי הקוד של תיאור קו (0-נקודה, שלילי-מרווח, חיובי-קו).

לדוגמא, נעבור על הגדרת **STARS**:

```
*angle,Angle steel

0, 0,0, 0,.275, .2,-.075

90, 0,0, 0,.275, .2,-.075

*grass,Grass area

90, 0,0, .707106781,.707106781, .1875,-1.226713563

45, 0,0, 0,1, .1875,-.8125

135, 0,0, 0,1, .1875,-.8125

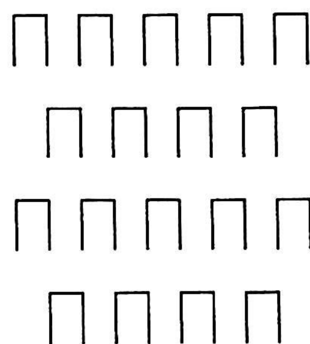
*stars,Star of David

0, 0,0, 0,.216506351, .125,-.125

60, 0,0, 0,.216506351, .125,-.125

120, .0625,.108253176, 0,.216506351, .125,-.125
```

מטלה : הגדר HATCH בשם NEWHATCH, בצורה כזו:



הגדרת TEXT FONTS

FONT וסמלים מוגדרים בקובץ מיוחד בעל סיומת **SHP**. הכנסת אינפורמציה לתוכו מתבצעת בעזרת EDITOR בפקודות מיוחדות.לפני טעינה ראשונה של ה-**FONT**,יש לבצע קומפילציה למבנה **SHX** (סעיף 7 בתפריט הראשי) במטרה לבצע את הטעינות במהירות ה-**FONT** נשמרים בצורה בינרית וקטורית, בזכרון ובדיסק, ובכך תופסים מעט מקום בזכרון.

מבנה SHAPE

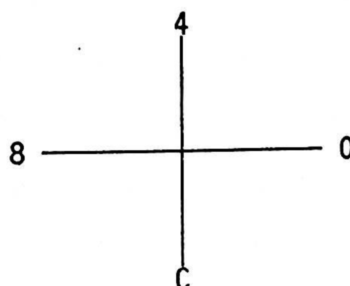
כל אחת מה-**SHAPES** מקבלת ערך בין **TEXT FONTS**.1-255 (השונים מ-**SHAPES** אחרים רק בהיותם סמלים לכל אות),דורשים מספרים המתאימים לטבלה ה- **ASCII**. ל-**SHAPES** אחרים ניתן לתת מספרים באופן שרירותי.

כל **SHAPE** מוגדר בשתי שורות:

שם הצורה , מספר BYTES בהגדרה, שם הצורה *

אם שם הצורה נכתב באותיות קטנות,הוא אינו נשמר ואינו תופס מקום.
בשורות ההגדרה מופרדות ההגדרות ע"י פסיקים.אפס מוביל מציין כתיבת **HEXADECIMAL** (10 = עשר , 010 = שש עשרה ב-HEXA).את ההגדרה מסיים תו - 0.

הגדרה אחת מחולקת לשני חלקים: כוון ואורך. כוונים מוגדרים לפי "שושנת הכוונים" בעלת 16 כוונים בעלי ערך HEXA.

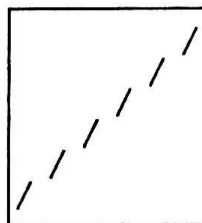


לדוגמא :

*230,6,BOX

014,010,01C,018,012,0

יוצר את הצורה הבאה:



גודל הממשי נקבע בזמן הכנסת הצורה בפקודת ה-SHAPE.

בנוסף לאלה, קיימת מערכת שלמה של קודים נוספים להגדרת ה- **FONT**.
חלקם מובאים כאן:

000 - סוף הגדרה

001 - PEN DOWN

002 - PEN UP

003 - חלק אורך וקטור במספר הבא.

004 - הכפל אורך וקטור במספר הבא.

005 - PUSH מיקום נוכחי.

006 - POP מחסנית.

007 - הכנס SHAPE קיים, בעל המספר הבא אחריו.

008 - תזוזה ב-XY הבאים, בשני המספרים הבאים.

009 - מספר רב של תזוזות, כמו 008 שנגמר ב 0,0

00A - שמינית קשת מוגדרת בשני המספרים הבאים.

00B - קטע קשת, מוגדר ע"י 5 מספרים הבאים.

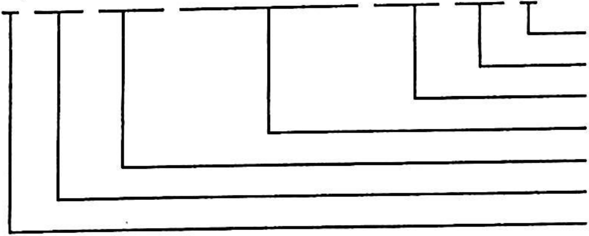
00C - קשת המוגדרת ע"י 3 מספרים הבאים.

00D - הרבה 00C.

00E - בצע פקודה הבאה, רק במקרה של כתב אנכי.

דוגמא לצורה CL:

```
*150,11,CENTERLINE
2,064,7,67,008,-12,-13,7,76,064,0
```



סיום ....

תזוזה 6 יחידות בכיוון 4 ....

הכנס צורה 76 ....

תזוזה ב-XY ....

הכנס צורה 67 ....

תנועה 6 יחידות בכיוון 4 ....

PENUP ....

התאמת ה-HELP

קובץ ה-HELP מנוהל לפי נושאים. הובא כאן חלק ממנו.ניתן להבחין כי הנושאים מופרדים למסכים בודדים ע"י \ בהתחלת שורה ושם נושא .

בנושא הדורש יותר ממסך אחד, תהיה שורת \ ללא נושא בראש הדפים הנוספים.

לקובץ זה ניתן להוסיף נושאים נוספים, כמו למשל אינפורמציה טכנית, גדלים סטנדרטיים או ספר טלפונים פרטי.

AutoCAD Command List (′ = transparent command)

APERTURE	CHANGE	DIVIDE	EXPLODE
ARC	CHPROP	DONUT	EXTEND
AREA	CIRCLE	DOUGHNUT	FILES
ARRAY	COLOR	DRAGMODE	FILL

\

AutoCAD Command List (′ = transparent command)

MOVE	PRPLOT	ROTATE	STRETCH
MSLIDE	PURGE	RSCRIPT	STYLE
MULTIPLE	QTEXT	RULESURF	TABLET
OFFSET	QUIT	SAVE	TABSURF

\APERTURE

The APERTURE command governs the size of the "target"

crosshairs

for object snap purposes.

Format: APERTURE

Object snap target height (1-50 pixels)

<default>: (number)

See also: Section 8.7 of the Reference Manual.

\ARC
The ARC command draws an arc (circle segment) as specified
by any of the following methods.

- three points on the arc
- start point, center, end point
- start point, center, included angle
- start point, center, length of chord
- start point, end point, radius
- start point, end point, included angle
- start point, end point, starting direction
- continuation of previous line or arc

3-point format: ARC Center/<Start point>: (point)

Center/End/<Second point>: (point)

End point: (point)

See also: Section 4.4 of the Reference Manual.

\AREA
The AREA command calculates the area and perimeter enclosed by a sequence of points you enter, or defined by a specified circle or polyline. A running total of measured areas can be kept, and you can ask AutoCAD to add or subtract subsequent areas from the total.

Format: AREA <First point>/Entity/Add/Subtract: (point)

Next point: (point)

Next point: (point)

Next point: ...press RETURN to end point entry

Area = nnnn Perimeter = nnnn

If add or subtract mode has been selected, AREA then adds to (or subtracts from) the running total, displays the running total, and repeats the options prompt. The options are:

Add - Adds the area to the running total.

Subtract - Subtracts the area from the running total.

Entity - Computes the area of a selected Circle or

RETURN - A null reply exits the AREA command

See also: Section 5.6 of the Reference Manual.

מטלה: בנה ספר טלפונים פרטי בקובץ ה-**ACAD.HLP** וגש אליו מתוך ה-ACAD.(בטל את קובץ ACAD.HDX כדי שהשינוי יקלט).

קבצי ה-ACAD

.$AC	קובץ עבודה זמני.
.BAK	קובץ השרטוט לפני השינוי האחרון.
.CFG	קובץ התצורה (מסך עכבר ועוד).
.DOC	תוספות אחרונות לתיעוד התוכנה,לאחר הדפסת הספר.
.DRV	DRIVER למסך.
.DVP	DRIVER למדפסת.
.DWG	קובץ השרטוט.
.DXB	DXF במבנה בינרי .
.DXF	בסיס הנתונים של ה-ACAD במבנה ASCII.
.DXX	ATTEXT במבנה DXF.
.ERR	הודעת שגיאה במקרה ש"עפים" מ-ACAD בצורה מבוקרת.
.EXE	התוכנית עצמה.
.FLM	קובץ הקישור מה-ACAD אל ה-ASHADE,כולל תיאור משטחים.
.HDX	INDEX עבור קובץ ה-HLP.
.HLP	קובץ HELP.
.IGS	תרגום השרטוט למבנה בתקן IGES .
.LIN	הגדרות קווים.
.LSP	תוכנת AUTOLISP.
.MNU	קובץ MENU - במבנה ASCII.
.MNX	קובץ MENU - אחרי קומפילציה.
.MSG	הודעת כניסה ל-ACAD.
.OLD	גירסה ישנה של שרטוט,לאחר תרגום לגירסה חדשה.
.OVL	חלק מתוכנת ה-ACAD.
.PAT	הגדרת HATCH.
.PGP	קובץ הגדרות פקודות חיצוניות מתוך ה-ACAD.
.PLT	קובץ שרטוט ל-PLOTTER.
.PRP	קובץ PLOT למדפסת ADI .
.SCR	קובץ SCRIPT (הוראות ACAD).
.SHP	הגדרת FONT או סמלים (SHAPE).
.SHX	הגדרה כנ"ל לאחר קומפילציה.
.SLB	SLIDE LIBRARY.
.SLD	SLIDE.
.TXT	קובץ ATTEXT או TEMPLATE,עבור ATTEXT (CDF/SDF).

<u>פקודות חיצוניות דרך קובץ ACAD.PGP</u>

ACAD מאפשר מנגנון של קריאה לרוטינות,מחוץ ל-ACAD מתוך התוכנית.הדבר מתבצע ע"י הגדרה בקובץ ACAD.PGP.

שורה בקובץ מגדירה את ההגדרות הבאות:

שם הפקודה ב-ACAD.

שם הפקודה עבור DOS.

נפח מקום בזכרון ה-ACAD אשר צריך להקצות לביצוע הפקודה.

PROMPT של ה-ACAD.

חזרה למסך ממנו יצאנו 4-מסך אלפאנומרי לאחר הביצוע 0.

```
CATALOG,DIR /W,30000,*Files: ,0
DEL,DEL,30000,File to delete: ,0
DIR,DIR,30000,File specification: ,0
EDIT,EDLIN,42000,File to edit: ,0
SH,,30000,*DOS Command: ,0
SHELL,,127000,*DOS Command: ,0
TYPE,TYPE,30000,File to list: ,0
```

<u>מטלה</u>: הוסף שורה לקובץ PGP, כך שתאפשר לפקודה MNU לצאת ל-EDITOR ולבצע עריכה לקובץ TEST.LSP ושורה נוספת לפקודה LSP יוצאת ל-EDITOR לעריכת קובץ TEST.LSP.

דרוש זכרון בסביבות 180000.

MENU

כללי

תפקיד התפריטים לאפשר לנו גישה ישירה לפקודות,הן על ידי חסכון בהקלדה והן בבנית תפריטים לקבוצות פעולות.התפריטים ניתנים להתאמה לציוד היקפי מסייע כגון:עכבר,מספרת,לוח מקשים מיוחד (AUX1).
מבנה התפריט נמצא בקובץ הקרוי XXX.MNU.
בקובץ נכללות פקודות ACAD,המקובצות בקבוצות תפריטים.

כדי לשנות\לבנות קובץ MENU,יש להכנס לתוכו בעזרת תכנית EDITOR.

תפריט פשוט

מטלה

הכנס לתוך קובץ TEST.MNU (בפקודת PGP MNU),את הפקודות הבאות:

תפריט פשוט ביותר יכלול פקודות המתחילות בעמודה 1:

```
LINE

ZOOM A

ZOOM W

GRID

ON

GRID ON
```

צא מה-EDITOR, חזור ל-ACAD

הפעל פקודת COMMAND : MENU TEST

מאחר ולא ציינו את סוג התפריט,ברירת המחדל היא תפריט SCREEN.

<u>תפריט עם כותרות</u>

לאותו תפריט ניתן להוסיף כותרות, כך שלא נשתמש כלל בשפת ה-ACAD.

<u>מטלה</u>

הכנס את הפקודות הבאות לקובץ TEST.MNU.
טען ה-MENU והפעל אותו.

```
[KAV]LINE
[ZOOMKOL]ZOOM A
[HALON]ZOOM W
[FINE ]SNAP 0.001
[COARSE ]SNAP 0.1
[G O N]GRID ON
[4PM-BYE]END
```

<u>פרקי התפריט</u>

ראשי הפרקים מצוינים על ידי שם שמור,ולפניו שלוש כוכביות.
תתי-סעיפים מצוינים על ידי שתי כוכביות.

```
***SCREEN
***POPn
***ICON
***BUTTONS
***TABLETn
***AUX1
```

לדוגמא:

```
***SCREEN
[HELP]HELP
[BYE]END
***TABLET1
[A-1]LINE
[A-2]ARC
[A-3]CIRCLE
***BUTTONS
ERASE
OOPS
```

אם הכותרת SCREEN*** לא מופיעה,ACAD יתייחס אל התפריט הראשון כאל תפריט זה.

תת תפריטים

תת תפריט מוגדר ע"י מילה כל שהיא,עם קידומת של **.
למשל:

**BOLTS

שמות חייבים להיות יחודיים בקובץ תפריט.כלומר,לא יהיו בתפריט ***SCREEN שני תת-תפריטי **BOLTS.

כל שורות התפריט אחרי שם התת תפריט עד לתת הבא,שייכים לאותו תפריט.
בתפריטי מסך ניתן להציג עד 20 שורות.

שורות תפריט שהופעל יחליפו שורות תפריט קודם,אלא אם יצוין בשם התת מספר השורה בה יתחיל התפריט.מעל שורה זו ישארו שורות התפריט הקודם.
לדוגמא: **SAMPLE 3.

יתחיל בשורה 3, וישאיר את שתי השורות הראשונות של התפריט הקודם לו ללא שינוי.

קריאה/הפעלת תתי-תפריטים:

תתי תפריטים ניתנים להתיחסות ע"י ציון תפריט האב שלהם,בצורה הבאה:
$SECTION=SUBMENU

כאשר אלה הקודים של התפריטים הראשיים:

S-SCREEN
P1-P10 POP
I-ICON
B-BUTTONS
T1-T4 TABLET
A1-AUX1

לדוגמא:

$S=PARTS

$T1=EDTCMDS

$T2=SCREEN

לפני שתת-תפריט מופעל, התפריט הנוכחי "נדחף" למחסנית. למשל, אם הפעלנו את $S=PARTS כדי לחזור לקודם, הדרך היא כדלקמן: =$S וחוזרים לתפריט הקורא.
גודל המחסנית הוא 8.
שיטה זו ניתנת להפעלה גם באמצע פקודה.

לדוגמא:

$S=ARCMENU ARC

ARC $S=ARCMENU

שתי הפקודות תבוצענה בצורה זהה. כאשר פקודת ARC תופעל,יופעל תפריט ARCSTUFF והפקודה תחכה לפרמטרים של פקודת ARC.

להלן דוגמא לקריאה מתפריט לתפריט:

<u>מטלה</u>: הכנס השורות הבאות לקובץ TEST.MNU

```
***SCREEN
[HELP    ]$S=HELP_ROOT

[BYE     ]END

**HELP_ROOT 2

[ FOR ]
[GENERAL]HELP ;
[ENTITIES]$S=ENTITIES_HELP
[DISPLAY ]$S=DISPLAY_HELP

[-MAIN-]$S=SCREEN
**ENTITIES_HELP 3
[LINE    ]HELP LINE
[CIRCLE ]HELP CIRCLE
[ARC     ]HELP ARC

[-PREV-]$S=
**DISPLAY_HELP 3
[ZOOM   ]HELP ZOOM
[PAN    ]HELP PAN
[VIEW   ]HELP VIEW

[-PREV-]$S=
***BUTTONS
;
REDRAW
```

<u>תפריטי POP</u>

אפשר ליצור עד 10 תפריטי POP, שיצויינו בשמות **POP1** עד **POP10**.

שורה ראשונה שם. ACAD בטעינה סורק את קובץ ה-MENU בחיפוש אחר POPnn*** ומרכיב את שורת הכותרת העליונה מכותרות תפריטי הPOP.

גם כאן ניתן ליצור היראכיה בין התפריטים, כאשר תפריט הקורא לאחר, וניתן לחזור לאחור בעץ ההיראכי.

לדוגמא:

```
***POP1
**P1A
[OPTIONS]
[ASHADE](LOAD "ASHADE") $P1=P1B $p1=*
[3D ROUTINES](LOAD "3D") $P1=P1C $P1=*

**P1B
[ASHADE]
[LIGHTS]^C^C$S=X $S=LIGHTS LIGHT
CAMERA]^C^C$S=X $S=CAMERA CAMERA
[ACTION]^C^C$S=X $S=ACTION
[--]
[EXIT]$P1=P1A

**P1C
[3D]
[CONE]^C^CCONE
[DISH]^C^CDISH
[DOME]^C^CDOME
[SPHERE]^C^CSPHERE
[TORUS]^C^CTORUS
[--]
[EXIT]$P1=P1A
```

טוען את תפריט X

טוען את תפריט LIGHTS ומבצע LIGHT

<u>תפריטי ICON</u>

כל פריט כולל שם ו-TEXT לביצוע.

שורה ראשונה שם ולא לביצוע. השם נרשם בתמונת ה-ICON מעל לתמונות.
תפריט ICON יכול לכלול עד 16 ברירות (יחד עם שורת הכותרת 17 שורות) שורות נוספות ACAD מתעלם מהן.

בתפריט ICON השמות מתייחסים לשמות SLIDES באותו מקום שבתפריטים אחרים מופיע הכותרת של הפעולה. כדי לחסוך את הצורך ביצירת SLIDES עבור הערות TEXT כמו EXIT או NEXT ניתן לכתוב TEXT ע"י הכנסת מרווח במקום הראשון בשדה השם (LABEL).

*=$I מציג תפריט ICON ומאפשר בחירה ממנו. גדלי ICON 4 9 או 16 תמונות.

דוגמא:

```
***ICON
**IPARTS
[ELECTRONIC PARTS]                                   שם התפריט
[CAP]^CINSERT CAP
[RES]^CINSERT RES
[NEON]^CINSERT NEON
[TRIODE]^CINSERT TRIODE
[ CANCEL ]^C
[FASTNERS]$I=FAST $I=*

**FAST
[FASTENERS]                                          שורת הכותרת
[NUT632]........
```

באותה מידה ניתן להפוך את ה-SLIDE-ים הבודדים בצורה הבאה:

```
***ICON
**IPARTS
[ELECTRONIC PARTS]                                   שם התפריט
[ELIB(CAP)]^CINSERT CAP
[ELIB(RES)]^CINSERT RES
[ELIB(NEON)]^CINSERT NEON
[ELIB(TRIODE)]^CINSERT TRIODE
[ CANCEL ]^C
[FASTNERS]$I=FAST $I=*
```

זאת כאשר ELIB הינה ספרית SLIDES.

<u>הכנת תפריט ICON</u>

1. הגדר תפריט ICON.
2. בנה השרטוטים ושמור אותם כ-SLIDES, ע"י MSLIDE.
3. בתוכנית SLIDELIB, בנה ספריה ע"פ המוגדר בתפריט .

<u>מטלה</u>: צור תמונת עיגול, ריבוע ומשולש.

בנה להם ספרית SLIDES. שמור אותם במקביל כ-BLOCKS.

בנה להם ICON, כך שיבצע INSERT ל-BLOCK כזה, כאשר בוחרים בו .

<u>פקודות הדורשות קלט</u>

```
[CIRCLE-1]CIRCLE \1
[LAYER-OFF]LAYER ? OFF \;
[MAKE RED]SELECT \CHANGE PREVIOUS ;PROPERTIES COLOR RED ;
```

כל הדוגמאות הקודמות מאפשרות רצף פקודות לפעולות תוך מתן אפשרות למשתמש להכניס פרטים לפקודה מסוג מרכז המעגל ,איזו LAYER לכבות או את איזה ישויות לצבוע אדום.

<u>שורת תפריט ארוך</u> - מסתיימת ב+ ואז ניתן להמשיך בשורה הבאה.

```
[BUBLE]*^C^CCIRCLE \24 ZOOM A SELECT L ;TEXT S STANDARD M @+
       18 0 \LINE @ \ ID MID,QUICK @ TRIM P ;ENDP,QUI @ ;+
      INSERT ARROW ENDP
```

חזרה על פקודה שנפסקת רק ע"י ^C מבוצעת על ידי * בתחילת שורת תפריט.

<u>הפעלת פקודות AUTOLISP</u> -

```
[BOX](SETQ A (GETPOINT "ENTER FIRST POINT CORNER: "));\=
(SETQ B  (GETPOINT  "ENTER OPPOSITE CORNER :"));\+
PLINE !A (LIST (CAR A) (CADR B)) !B (LIST (CAR B) (CADR A))+
C;
```

<u>מטלה</u> : כתוב MACRO, המופעל ע"י פקודת MENU, המבצע ציור מעגל (מאפשר לקבוע את הרדיוס ומיקום המרכז) ומחלק אותו בעזרת פקודת DIVIDE (דאג לשמירת BLOCK כלשהוא לסימון).

<u>מבנה קובץ DXF</u>

מידע שנמצא בבסיס הנתונים של ה-ACAD, ניתן לגישה ישירה על ידי פעולות ALISP מתוך ה-ACAD, או לגישה ע"י תוכנות חיצוניות ל-ACAD, דרך קבצי DXF הנמצאים בפקודת DXFOUT.

קבצי DXF הינם קבצי ASCII רגילים, על פי מבנה המוגדר ונתמך ע"י AUTOCAD ותוכנות צד שלישי.

פקודת DXFOUT כותבת קובץ בעל סיומת DXF. הפקודה מאפשרת לבחור איזה ישויות תכתבנה החוצה, תחת אופציה של ENTITIES.

<u>מטלה</u>: צייר כמה קוים ב-AUTOCAD והוצא אותם בפקודת DXFOUT, פעם אחת בעזרת אופצית ENTITIES ופעם שניה - את כל השרטוט.
בדוק הקובץ בעזרת ה-EDITOR.

<u>מבנה כללי:</u>

1. כותרת - **HEADER** - אינפורמציה כללית לגבי השרטוט - פרמטרים, משתני מערכת וערכם.

2. פרק הטבלאות - **STYLE LTYPE LAYER** טקסטים UCS VIEW.

   DWGMGR VPORTS - אינו בשימוש כרגע.

3. פרק ה-**BLOCKS**.

4. פרק ה-**ENTITIES**.

5. **END OF FILE**.

הקובץ מכיל אוספים רבים של קבוצות, הכוללות כל אחת זוגות של שורות. שורה ראשונה הינה קוד קבוצה, שהוא מספר חיובי שלם שונה מאפס. השורה השניה הינה ערך, התלוי בסוג הקבוצה.

0-9 שרשרת STRING.

10-59 FLOATING POINT.

60-79 INTEGER.

210-239 FLOATING POINT.

999 COMMENT.

<u>קודים לקבוצות</u>:

0	תחילת ENTITY, טבלה.
1	ערך המתאר ENTITY.
2	שם ,ATTRIBUTE,שם BLOCK וכו'.
3-4	שמות אחרים.
5	.ENTITY HANDLE
6	.LTYPE NAME
7	.TEXT STYLE NAME
.	
.	
.	
10	קורד X עיקרי (תחילת קו).
.	
.	
.	
20	קורד Y עיקרי.
	וכו'.

לדוגמא: DXF של שרטוט קו וקשת, בהוצאת ה-ENTITIES בלבד.

```
  0                          תחילה
SECTION
  2                          שם
ENTITIES
  0                          תחילה
ARC
  8                          LAYER
0
  5                          HANDLE
5
 10                          X CORD
5.948536
 20                          Y CORD
3.002092
 30                          Z CORD
0.0
 40                          רדיוס
2.600387
 50                          זוית מ
343.481001
```

```
 51                                  זוית ל
162.070342
  0                           ENTITY תחילת
LINE
  8                                  LAYER
0
  5
4
 10
3.242807
 20
5.311096
 30
0.0
 11
10.783621
 21
4.274018
 31
0.0
  0
ENDSEC
  0
EOF
```

מאפייני BLOCKS - BLOCK ATTRIBUTES

מאפיין **BLOCK** הינו ישות ACAD המתארת את הבלוק. היא ניתנת להצגה על המסך וקרויה **ATTRIBUTE TAG**. אחרי שהיא מוגדרת, ניתן לצרפה להגדרת בלוק ואז בכל **INSERT**, ACAD ידרוש לערך המאפיין. ערך זה יאוחסן כשרשרת **STRING** של תוים. ערך כזה ניתן להוציא מהשרטוט לקובץ חיצוני, כדי להשתמש בו בתוכנות אחרות חיצוניות ל-ACAD, כמו **DBASE** או **LOTUS**.

קיימת סדרה שלמה של פקודות, המטפלות במאפייני בלוקים:

ATTDEF - מגדיר מאפיין.

ATTEXT - מוציא מאפיינים של בלוקים בשרטוט לקובץ חיצון.

ATTEDIT - מאפשר גישה למאפיינים, לצורך שינוי ערכם.

ATTDISP - מאפשר לשנות את מצב ההצגה של המאפיינים.

דוגמא:

נניח מקרה של מנהל נכסים בחברה משרדית, המעונין להכניס מספר שולחנות לשרטוט במשרד. כל שולחן מיועד לעובד בחברה וכמובן שמו יהיה רשום על השולחן בשרטוט. במקרה שאין השולחן שייך לעובד מסוים,נרצה שיהיה רשום **"CLARK"** על השולחן.

נשרטט מלבן, שיהיה בלוק-השולחן בשרטוט.

נפעיל פקודת **ATTDEF**

ונגדיר מאפיין **TAG** בשם **EMPLOYEE**, עם ברירת מחדל **CLARK**.

בשלב הבא, נבצע **WBLOCK** למלבן ולמאפיין.

פקודת **INSERT** לבלוק-השולחן משרטטת אותו ודורשת מתן ערך לשם העובד.

כאשר גמרנו את השרטוט, ניתן לקבל רשימת מאפיינים לכל בלוק, כולל מיקום הבלוק בשרטוט ושם העובד .

<u>הגדרת הבלוק:</u>

ATTDEF INVISIBLE,CONSTANT,VERIFY,PRESET

ENTER (ICVP):

ATTRIBUTE TAG: EMPLOYEE

ATRIBUTE PROMPT: EMPLOYEE NAME?

DEFUALT VALUE: CLERK

לאחר הגדרת המאפיין, יש לרשום **WBLOCK** לבלוק בשם **DESK** ולמאפיין.
ואז, בזמן **INSERT**, באה השאלה:

ENTER ATTRIBUTE VALUES

EMPLOYEE NAME? <CLERK> :

כאשר ממלאים את השרטוט בבלוקים כאלה, ניתן להוציא קובץ **ASCII** במבנה הבא:

DESK 110.0 250.0 CLERK

DESK 150.0 300.0 JOHN SMITH

DESK 200.0 200.0 CAROL WHITE

הקובץ מוצא בצורה הבאה:

יש ליצור תחילה קובץ "מסכה" שיסמל את מבנה השורה (רשומה) הבודדת. במקרה זה הקובץ ייקרא **TEMP.TXT** (סיומת TXT מתחייבת).

BL:NAME	C005000	שם הבלוק
BL:X	N005001	קואורדינה עם ספרה אחת אחרי הנקודה
BL:Y	N005001	
EMPLOYEE	C010000	

בשורה למעשה, מתוארים שדות האינפורמציה ומבנה הכתיבה שלהם.

כדי לכתוב את הקובץ, יש להפעיל פקודת **ATTEXT**:

ATTEXT
CDF,SDF, OR DXF ATTRIBUTE EXTRACT (OR ENTITIES)? <C>

TEMPLATE FILE NAME:

EXTRACT FILE NAME: <ברירת המחדל שם השרטוט>

<u>מטלה:</u>

בצע את הדוגמא, כאשר בנוסף לשם העובד על כל שולחן תהיה גם אינפורמציה על שם היצרן, שנת הרכישה וערך השולחן בזמן הרכישה.
הכן קבצי מסכה עבור הוצאת סה"כ ערכי השולחנות, מיקום השולחנות ושנת הרכישה שלהם, מיקום השולחנות ע"פ יצרן לצורך הדרכת טכנאי של היצרן.
שם העובד המשתמש בשולחן יהיה - EMPLOYEE.
ערך השולחן בזמן הרכישה יהיה - VALUE.
שנת ייצור של השולחן יהיה - MYEAR.

שפת AUTOLISP

מטרות:

נושאים נכללים:

- קשר בין AUTOCAD ל-AUTOLISP.
- הגדרת בעיה ותכנון תוכנית.
- תיעוד ואחזקה.
- תהליך ה-EVAUATION ב-ALISP.
- מבני אינפורמציה ב-ALISP.
- פעולות אריטמתיות.
- השמת משתנים.
- הגדרת פונקציות.
- משפטי תנאי.
- לולאות.
- גישה לישויות בדיקה ושינוי.
- גישה לטבלאות.
- קלט פלט ב-ALISP.

ידע נדרש:

- הכירות טובה עם פקודות AUTOCAD.
- הכירות עם אפשריות תפריטים,קובץ PGP., SCR., LIN. וכו.
- ידע בהפעלת EDITOR TEXT (למשל EDLIN או NE).

תכונות ה-ALISP

- פשוטה להבנה, קלה ללימוד דקדוק קשיח ולא מורכב.
- עובדת בשיתוף עם ה-DRAWING EDITOR של ה-AUTOCAD, כך שניתן להציג את תוצאות החישובים בשרטוט ה-ACAD.
- שפת ה-ALISP הותאמה לשליטה והפעלה של שפת ה-AUTOCAD.
- שגרות ה-ALISP נבנות כבלוקים פשוטים, הניתנים לחיבור למבנה מורכב ככל הנדרש.
- שפת ה-ALISP קלה לתיעוד ולתחזוקה.
- השפה מאפשרת איטרציות (לולאות), משפטי תנאי ורקורסיה.

מטרת ה-ALISP:

שפת ה-**ALISP** הינה הכלי החזק ביותר בתוכנת ה-**AUTOCAD** להתאמה לצרכי המשתמש. פשוטה מחד גיסא ובעלת עוצמה גדולה לביצוע מטלות בבסיס הנתונים של ה-**ACAD**, מאידך גיסא.

ישומים רבים ומגוונים נכתבו ב-**ALISP** בנושאים, כמו **NC**, **AEC**, מתן מידות מיפוי ועוד.

שפת ה-**ALISP** מטפלת ברשימות. במקרה של ה-**ACAD**, אלה אוסף הטבלאות והישויות המרכיבות את בסיס הנתונים ממנו מורכב השרטוט.

ה-**ALISP** כלול בכל תוכנת **AUTOCAD** .

AUTOLISP

ATOM AND LISTS

ATOM - הינו פריט המידע הפשוט ביותר ב-LISP. לחילופין, כל פריט שאינו LIST הינו ATOM.

דוגמאות:

1	.INTEGER
487.284	.REAL
"THE QUICK BROWN FOX "	.STRING
+	.FUNCTION
XXX	.SYMBOL

אוסף של ATOMS יהווה רשימה LIST.

.(+ 3 1)

LIST - כל מידע ב-**ALISP** הינו **ATOM** או **LIST**. (**LIST** יכול לכלול בתוכו **ATOM** ו\או **LIST**).

(+ 1 3)

יש להקפיד על זוגות מאוזנים של סוגריים ימניים ושמאליים.

דוגמא להודעה על כך שחסר סוגר ימני: 1>

הפתרון לכך 1>)

אפשרות נוספת לפתרון 1>")

פריטי רשימה יהיו מופרדים אחד מהשני, במרווח אחד או יותר.

ALISP יבדוק כל **LIST** וינסה לקבל עבורה ערך.

(1 3 +) - יתן ערך 4.

((1 2 +) 2 +) - יתן ערך 5.

לכאורה, ניתן להניח כי ה-**ALISP** מבצע הערכה החל מהסוגריים הפנימיים ביותר, למרות שלמעשה אין זה כך.

NIL - מקרה מיוחד. NIL הינו **SYMBOL**, המוכר גם כ-**LIST** וגם כ-**ATOM**. סימון של () הינו NIL ומשמעותו "אין".

FUNCTIONS - הינן פקודות מוגדרות מראש. הן יכולות להיות פעולות אריטמתיות, או פלט-קלט. הן חלק מ-**LIST** אך לא כל **LIST** כולל **FUNCTION**. בפונקצית ALISP TERPRI מדפיסה למסך שורה.

COMMAND : TERPRI

יתן הודעת שגיאה .

COMMAND : (TERPRI)

לעומת זאת יתבצע נכון.

ובאותה דרך: (1 3 +) או (9 5 -) .

ה-**FUNCTION** מופיעה תמיד ראשונה ב-LIST. האברים הבאים הינם ה-**ARGUMENTS** לפונקציה.

ARGUMENT יכול להיות גם הוא **LIST** - ((2 1 +) 1 +)

EVALUATION

הפקודות ש-ALISP מעריכה בעזרתן LIST הינן חלק מה-LIST עצמו. כלומר, כל LIST כוללת את ההוראות של עצמה.

AUTOLISP מעריכה את ה-LIST בשיטה הבאה:

(+ 1 2)

ה-AUTOLISP עוברת על ה-LIST מגלה את ה +. מאחר וה-ALISP מצפה ל-FUCTION כאלמנט ראשון, היא מזהה את ה + כ-FUNCTION חוקי, וממשיכה לבצע את פעולת ה-FUNCTION +. במקרה שה + לא היה FUNCTION חוקי, היתה ה-ALISP משחררת הודעה: ERROR BAD FUNCTION, בצירוף הביטוי בו הוא מופיע.

הערך המוחזר מבדיקת ה-FUNCTION הינו קבוצת הפעולות המציינות את המשך עיבוד ה-LIST. במקרה זה, ההוראה היא: עבור על כל שאר ה-ARGUMENTS של ה-LIST, אגור את ערכיהם. בהגיעך לערך האחרון - (סוגר ימני), חבר את כולם והחזר את התוצאה:

.(+ 1 2 3 4 5)

אי לכך, מחזירה ה-ALISP ערך של 3 (INTEGER) בפעולה (2 1 +).

שיטה זו - בה מציבים את ה-FUNCTION תחילה, קרויה PREFIX NOTATION.

באותה צורה, נבחן גם הביטוי המורכב יותר: ((2 1 +) 3 +), כאשר במקרה זה מזוהה שוב + כ-FUNCTION 3 מאוחסן ואז ה-ALISP פונה להערכת (2 1 +), כאשר שוב מזוהה תחילה ה +, ובסוף ההערכה מוחזר הערך 3 ומתווסף לערך 3 הקודם, ולביטוי כולו יש ערך של 6.

ערך המוחזר מפעולת פקודת ACAD יהיה תמיד NIL, אך הפעולה עצמה תתבצע ב- AUTOCAD לדוגמא:

```
( COMMAND "LINE" PAUSE PAUSE  "" )
```

מחזיר NIL, אך מבצע קו.

```
( SETQ PT1 (GETPOINT "\nCENTER OF CIRCLE " ))

(COMMAND "CIRCLE" PT1 1.5 )
```

<u>**השמת משתנים**</u>

הכנסת ערך למשתנה ב-ALISP קרויה השמה או **BINDING**.

הדבר נעשה ע"י מתן ערך לשם **SYMBOL**. פקודת **SETQ** הינה הדרך העיקרית לביצוע השמה.

כדי להכניס 1235.5 למשתנה AAA, נכתוב: (SETQ AAA 1235.5).

הערך שמוחזר מפעולה זו הוא הערך 1235.5.

כדי לבדוק את ערך AAA, נכתוב: AAA!.

<u>**תרגיל :**</u>

הדפס את הביטויים הבאים והשווה את התוצאות.

```
(SETQ A 15.5)

!A

(SETQ B (+ 5. 4. 9. 3.))

!B

(* A 2.0)

(/ B 2.0)

(+ (/ B 3.0) A)

(SETQ C (+ (/ B 3.0) A))

!C

(SETQ D (/ (+ A B) 2))

!D
```

השמה הידברותית - INTERACTIVE BINDING

פקודת **SETQ** מאפשרת הכנסת ערך למשתנה. ניתן לבצע זאת בצורה הידברותית, כאשר השיגרה מבצעת **PROMPT** (דרישה) למשתמש.

לדוגמא:

```
(SETQ PT1 (GETPOINT) )
```

מאפשרת הכנסת קואורדינטת נקודה לתוך משתנה PT1.

GETPOINT עוצר את מהלך התוכנית, עד שהמשתמש מכניס קואורדינטות נקודה, אם בעזרת הצבעה בעכבר ואם בהקלדת שניים או שלושה ערכים, המופרדים ביניהם בפסיקים (X,Y,Z).

כאשר מוכנסת נקודה, ערכה נכנס כ-**LIST** לתוך משתנה PT1.

כך התבצעה פעולת **השמה-הידברותית**.

יש לשים לב, כי ה-**GETPOINT** לא הוציא כל הודעה למשתמש. נוצר פוטנציאל לבעיה, משום שמשתמש המפעיל את התוכנית לא יידע מה נדרש ממנו, או לחילופין- מדוע עצרה פעולת התוכנית או- למה היא ממתינה.

לכן, מומלץ להכניס תמיד הודעה המציינת למפעיל למה מצפה התוכנית ממנו. הדרך הנכונה לרשום זאת תהיה :

```
( SETQ PT1 (GETPOINT "\nENTER POINT :"  ))
```

n\ מציין קפיצה לשורה חדשה.

קלט סוגי אינפורמציה שונים

ALISP כוללות פקודות שונות, לקלט סוגי אינפורמציה שונים:

נקודה		GETPOINT
מרחק		GETDIST
מספר שלם	(INTEGER)	GETINT
מספר ממשי	(REAL)	GETREAL
מחרוזת תווים	(STRING)	GETSTRING

לדוגמא:

```
(SETQ X (GETREAL "\nENTER A REAL NUMBER PLEASE") )
```

מבקש מהמשתמש להכניס ערך מספרי ממשי.

X! - מאפשר למשתמש לבדוק את הערך שהוכנס.

או :

```
(SETQ NAM (GETSTRING "\nPLEASE ENTER YOUR NAME" ))
```

NAM! - מאפשר למשתמש לבדוק את ערך המשתנה.

אם היינו רוצים לאפשר למפעיל להקיש את שמו, כולל **מרווחים** בין השם הפרטי לשם המשפחה, היתה הפקודה נראית כך:

```
(SETQ NAM (GETSTRING T "\nPLEASE ENTER YOUR NAME" ))
```

לקבלת מרחק או אורך - **GETDIST**.

```
(SETQ DST (GETDIST "\nPLEASE ENTER DISTANCE "))
```

את המרחק ניתן לציין ע"י הקלדת מספר או ע"י ציון שתי נקודות בעזרת עכבר.

!DST - ערך המשתנה.

את הפקודה ניתן גם לבצע תוך הגדרת נקודת יחוס ממנה מודדים מרחק:

```
(SETQ DIST (GETDIST (LIST 0 0) "\nPLEASE ENTER DISTANCE"))
```

התוכנה יוצרת "חוט גומי" לנקודת היחוס. במקרה זה 0,0.

<u>תרגיל:</u>

הכנס למשתנים הבאים ערכי נקודות בעזרת פקודות **SETQ** ו-**GETPOINT**. ובדרך בחירת נקודות בעכבר.

```
PT1     1,1
PT2     11,1
PT3     11,8
PT4     1,8
```

הדפס הביטויים הבאים:

```
(COMMAND "LINE" PT1 PT2 PT3 PT4 "C")

(COMMAND "LINE" PT1 PT3 "")

(COMMAND "LINE" PT2 PT4 "")

(COMMAND "CIRCLE" "2P" PT1 PT3)
```

טען את ה-STRING לתוך המשתנים:

GETSTRING בפקודת

```
ST1     "FIRST LINE"
ST2     "SECOND LINE"
```

הדפס הביטויים:

```
(SETVAR  "TEXTEVAL"  1)

(COMMAND "TEXT" PAUSE PAUSE 0 ST1)

(COMMAND "TEXT" "" ST2)
```

חישובים אריתמטיים

חישובים מבוצעים ב-ALISP ב-LIST, כאשר התוצאה היא ערך הביטוי.
שני סוגי נתונים משתתפים בחישובים :

שלמים: INTEGER - מספר שלם בין 32768 ל-32767-.

ממשיים: REAL - מספר כל שהוא, החייב לכלול נקודה עשרונית.

המספרים הממשיים מיוצגים במחשב בדיוק של 15 ספרות אחרי הנקודה .

מספרים קטנים מ-1 חייבים 0 מוביל.

כדי לקבל תוצאת מספר ממשי, חייב אחד האופרנדים להיות ממשי.

<u>תרגיל:</u>

הדפס את הביטויים הבאים :

```
(+ 1 3)

(+ 1 (+ 1 3))

(- 9 5)

(/ 8 3)

(/ 8. 3)

(+ 15. (/5. 3.))

(+ 4 0.75)

(+ 4 .75)

(* 3. 9.)

(- 24. 3. 16.5)

( + 7. -3. 4. -2.)

(SQRT 25.)

(EXPT 6.0 2.0)
```

בצע הפעולות החשבוניות:

3+10+5

20 X 15

16 - 10

15 / 4

(5 + 10) X 2

5 + (10 X 2)

קשר בין AUTOLISP ל-AUTOCAD

ה-ALISP מאפשרת הפעלת פקודות AUTOCAD. ה-ALISP מסוגלת לספק קלט לרמת ה-COMMAND של AUTOCAD, תוך העברת נתונים שהיו אגורים במשתנים או שחושבו קודם לכן. בנוסף יכולה ה-ALISP לקבל מידע, ישירות מבסיס הנתונים של ה-AUTOCAD לגבי השרטוט.

לדוגמא, הפעלת פקודת ACAD: יש ליצר מטריצה של 3X5 של הישות האחרונה, כאשר המרחק בין השורות הוא 11 יחידות חלקי 6, והמרחק בין הטורים הוא 1.45 .

(COMMAND "ARRAY" "LAST" "" "R" 3 5 (/ 11.0 6) 1.45)

יתרון שיטה זו על פקודות מקרו, בולט בזמן החישוב של 11 חלקי 6.

פריטי המידע בחישובי ALISP יכולים לכלול כל הנתון מחישוב אחר נתון בשרטוט, או נתון המצוי בקובץ ASCII אחר במחשב.

ALISP יכולה לבצע שאילתות ישירות לגבי נתונים בבסיס הנתונים של השרטוט וכן לשנות נתונים באופן ישיר בבסיס הנתונים של השרטוט. השרטוט על המסך מתעדכן בהתאם, בו זמנית.

שימוש בפקודות AUTOCAD דרך ה-AUTOLISP

כל פקודות AUTOCAD, כמו ERASE, LINE, ZOOM וכו', יכולה להיות מבוצעת ע"י ALISP.

(COMMAND "LINE" PAUSE PAUSE "") או (COMMAND "REDRAW")

(COMMAND "ZOOM" "E")

"" מציין ENTER לסיום פקודה.

ניתן להשתמש ב-(COMMAND) כ-C^ לביטול פקודה, באמצע הביצוע.

גישה ישירה לבסיס נתוני ACAD

ALISP יכולה לבצע שאילתות ישירות למשתני ACAD. לדוגמא:

(GETVAR "SNAPMODE") מחזיר את ערך משתנה SNAPMODE.

באותה מידה, ניתן לטעון ערכים למשתני ACAD:

(SETVAR "LUNITS" 2)

אותו דבר יכול להתבצע ע"י פקודת ACAD - SETVAR.

<u>תרגיל</u>

השתמש בפקודת **SETVAR**, על מנת לקבל את ערכי המשתנים הבאים:

ACADVER	SNAPUNIT
CLAYER	GRIDUNIT
DWGNAME	SNAPMODE
DWGPREFIX	DIMSCALE
MENUNAME	LTSCALE
LASTPOINT	FILLETRAD
LASTPT3D	ORTHOMODE

הכנס בעזרת **ALISP**, ערכים למשתנים הבאים:

SNAPMODE	1
DIMSCALE	12
LTSCALE	12
FILLETRAD	0.25

<u>סוגי נתונים, מידע ומשתנים</u>

סוגי מידע, בהם תומכת ה-ALISP, הינם:

LISTS.

SYMBOLS.

STRINGS.

REAL NUMBERS.

INTEGERS.

FILE DESCRIPTORS .

AUTOCAD ENTITY "NAMES".

AUTOCAD SELECTION SETS .

SUBR (BUILTIN FUNCTIONS).

FUNCTION PAGING TABLE.

<u>LISTS</u>

((SETQ X '(1 2 3), מחזיר (3 2 1) ומכניס ערך זה למשתנה X.

(3 2 1), היא **LIST** וכן גם הפקודה כולה.

הערה : הגרש ' מורה ל-ALISP לא לבצע פעולת הערכה **EVALUATION** על הרשימה (3 2 1), אלא להעבירה כמות שהיא.

LISTS מוערכים לפי האלמנט הראשון ב-**LIST**.

<u>SYMBOLS</u>

(SETQ X 'A) מחזיר A.

מכניס את ה-SYMBOL A לתוך משתנה X.

!X מחזיר A.

הערה: הגרש לפני ה-A מציין ל-**ALISP** לא לבצע הערכה ל-**SYMBOL A**, אלא לטעון אותו ל-X.

SYMBOLS מוערכים לפי ערכם הנוכחי.

STRINGS

"THIS IS A STRING" מחזיר, (SETQ X "THIS IS A STRING")

וטוען זאת למשתנה X.

STRINGS מוערכים לעצמם.

REALS

(SETQ X 45.321) מחזיר ערך של 45.321, וטוען את X.

X! מחזיר ערך 45.321 .

REALS אינם מוערכים, אלא נלקחים בערכם המספרי.

INTEGERS

מספר שלם, בין 32768- ל 32767 המצוין בלי נקודה עשרונית, הינו INTEGER. ALISP מעריכה פעולות חשבוניות בהם משתתפים רק מספרים שלמים בתוצאה של מספר שלם.

(/ 27 3) -------> 9 הגיוני....

(/ 28 3) ------> 9 פחות הגיוני..

מאחר ובביטוי השני משתתפים מספרי INTEGER בלבד, התוצאה היא INTEGER והשארית מקוצצת. לעומת זאת:

(/ 28 3.) --------> 9.3333

INTEGERS אינם מוערכים, אלא נלקחים בערכם המספרי.

FILE DESCRIPTORS

ALISP כוללת גם פקודות קלט ופלט, המאפשרות לקרוא ולכתוב לקבצים חיצוניים.

הדבר מתבצע בשלושה שלבים:

- תחילה יש לפתוח הקובץ בפעולת OPEN.
- לאחר מכן כותבים, או קוראים, READ WRITE.
- ולבסוף יש לבצע סגירה CLOSE.

פעולת הפתיחה מחזירה הצבעה לשם הקובץ, שהוא ה-FILE DESCRIPTOR.

(SETQ X (OPEN "TESTFILE" "W")) מחזיר בסגנון<FILE: #FE5D>

הפקודה פותחת קובץ בשם TESTFILE על הדיסק, ומכינה אותו לכתיבה.
בגישות הבאות לקובץ, יש לציין את אותו ערך שה-OPEN מחזיר - במקרה למשתנה X.

בתוך X נמצא FILE DESCRIPTOR.

ENTITY NAMES

AUTOCAD נותן לכל ישות בשרטוט שם, המתקיים בכל זמן ההפעלה שלו.

שמות אלה אינם מאוחסנים בשרטוט השמור (.DWG)

ALISP יכולה לגשת לכל ישות, לקרא אותה או לשנותה.

ENTITY NAMES מאפשרים בדיקה ו\או שינוי, בכל פריט בבסיס הנתונים של השרטוט.

(SETQ X (ENTNEXT)) ------> <ENTITY NAME: 60000C0>

!X מחזיר שם ישות.

שמות ישות ENTITY NAMES מוערכים לעצמם.

<u>SELECTION SETS</u>

ALISP כוללת סדרת פקודות, המטפלות ב-SELECTION SETS.

SELECTION SET כולל בתוכו שמות קבוצת ישויות, אשר בעזרת פקודות מתאימות ניתן ליצור , לבדוק, להוסיף ולשנות את כולן בבת אחת.

```
<SELECTION SET : 1>    <------- (SETQ X (SSGET "X"))

<SELECTION SET : 1>    <----------      !X
```

X כולל את שמות הישויות שנבחרו.

SELECTION SET מוערכים לעצמם.

<u>**SUBRS**</u>

FUNCTIONS הבנויות בתוך ה-**ALISP** כמו +, או **SETQ**, או **OPEN**, הינן סוג נתונים מיוחד הקרוי **SUBRS**.

```
!SETQ    <--------    <SUBR: #4A2>
```

או מספר דומה.

מספר זה הינו ההצבעה (POINTER) הפנימי ב-**ALISP** לפונקצית ה-**SETQ**.

כל אחת מפקודות ה-**ALISP** בספר ה-**APR**, הינה **SUBR**.

<u>**פונקצית ה-TYPE**</u>

פונקצית **TYPE** ב-**ALISP** מחזירה את סוג המשתנה, על פי הסוגים שנסקרו לעיל.

<u>תרגיל</u>

הדפס את הכתוב.

על מנת להפעיל את ENTNEXT ו-SSGET, צריך שתהיה ישות משורטטת.

Expression	Vaiue of the expression	
(setq x 1)		
!x	1	
(type x)	INT	Integer
(setq x 'a)		
!x	A	
(type x)	SYM	Symbol
(setq x 1.0)		
!x	1.000000	
(type x)	REAL	Real number
(setq x "hello.")		
!x	"Hello."	
(type x)	STR	String
(setq x '(1 2 3))		
!x	(1 2 3)	
(type x)	LIST	List
(setq x (open "test" "w"))		
!x	<File: #number>	
(type x)	FILE	File descriptor
(close x)	nil	
(setq x (entnext))		
!x	<Entity name: number>	
(type x)	ENAME	Entity name
(setq x (ssget "x"))		
!x	<Selection set: number>	
(type x)	PICKSET	Selection set
(type setq)	SUBR	Internal function
!setq	<Subr: #number>	

תיעוד תוכנה, הערות ואחזקה

אחד הצעדים הראשונים בתחילת תכנות ALISP הוא **תעוד התוכנית.**

דאג תמיד, לתיעוד התוכנית, ללא יוצא מהכלל !!!

קיימות לכךי סיבות כבדות משקל, והחשובה מבינהן מתמצית בשאלה:

איזו מבין שתי התוכניות קלה יותר להבנה ?

תוכנית מס. 1

```
(DEFUN X (XX)  (/ (* XX PI)  180 ))
```

תוכנית מס. 2

```
;FUNCTION DTR
;DEGREES TO RADIANS
;CONVERTS ANY NUMERIC VALUE FROM DEGREES TO ITS EQUIVALENT
;       VALUE IN RADIANS.
;
;FORMAT   (DTR <number> )
;EXAMPLE  (DTR 180)  RETURNS  3.141579
;WRITTEN BY PHIL CARTESIAN JANUARY 5 , 1988
;
(DEFUN  DTR (D)
                (/ (* D PI ) 180 )
)
;
;
```

שתי התוכניות מבצעות בדיוק אותה פונקציה - הפיכת מעלות לרדיאנים.

הראשונה אינה ניתנת להבנה ואילו השניה קלה להבנה, לשימוש ולאחזקה.

תוכניות עשויות להיהפך מורכבות ובלתי מובנות לאחר זמן, אפילו לאותם אנשים אשר כתבו אותן. ביטויים שהיו פשוטים ומובנים בעת הכתיבה, הופכים לסיוט לא מובן חודש מאוחר יותר, כאשר יש לתקן BUG או לבצע שינוי כלשהוא.

תוכניתנים מיומנים **מתעדים** כל תוכנית שנכתבה על ידם, מפזרים בתוכניות הערות מסבירות אחר כל מספר שורות תוכנה. הם מספקים למשתמש תיעוד אודות הקלט הדרוש לתוכנית , על מה שהתוכנה מבצעת עם הקלט ועל מהות הפלט. לעיתים מומלץ לציין באיזה שיטה נפתרה הבעיה ואף לציין מקורות (ספרות מקצועית) בהם ניתן למצוא סימוכין לפתרון זה.

תוכניות ALISP מאפשרות הערות בסימן ; בתחילת ההערה.

<u>שימוש בנקודה פסיק:</u>

```
 (* 180.0 3.14159)      ;THIS IS A COMMENT


;DIVIDE X BY Y AND
; SUBSTRACT RESULT BY Z

(- (/ X Y ) Z )
```

כל הערה מתחילה ב- ; וממשיכה עד לסוף אותה שורה.

<u>כתיבת הערות בדרך לא נכונה:</u>

```
; (* 180.0 3.1415)    ; THIS A COMMENT


;DIVIDE X BY Y AND
 SUBTRACT IT FROM Z

(- ( / X Y ) Z )      ; ASSUME X AND Y ARE
                        NUMERICS.
```

בנית LISTS (רשימות)

LIST-ו QUOTE

נקודה ב-AUTOCAD מיוצגת ע"י רשימה של שני מספרים ממשיים.

נקודה 3,5 תיוצג על ידי (3.000000 5.000000) .

ניתן לבנות ב-ALISP רשימות דומות לזו, כך ש-AUTOCAD יכיר בהן כנקודות.

פונקציות LIST ו-QUOTE מנוצלות ליצירת רשימות כאלה.

LIST לוקחת כל מספר של אברים, ובונה מהם רשימה **LIST**.

QUOTE מחזירה ביטוי בלי להעריך (**EVALUATE**) אותו. במילים אחרות, הפקודה **QUOTE** עוצרת את ה-ALISP מלהעריך את הביטוי.

במקום פונקצית **QUOTE** ניתן לרשום ׳. באותה מידה שבה ניתן לרשום () במקום **NIL**.

תרגיל

הדפס את הביטויים הבאים:

```
(SETQ A 5.0)
(SETQ B 6.0)

(LIST 3. 4.)
(LIST A B)
(SETQ PT1 (LIST A B))
!PT1

(SETQ BIGLIST (LIST 3 2 1 0))

(QUOTE (2. 3.))
(SETQ PT2 (QUOTE (8. 9.)))
(SETQ PT2 '(8. 9.))
(SETQ PT3 '(A B))

!PT2
!PT3

(SETQ C 'A)
!C
```

הבדלים בין QUOTE לבין LIST

נרשום :

```
( SETQ A 5.0)
( SETQ B 6.0)
```

שים לב להבדל בין
לבין

```
( LIST A B)
(QUOTE (A B))
```

```
( LIST A B)      ---------> (5.00000 6.00000)
(QUOTE (A B))    ---------> ( A B )
```

QUOTE הינה פעולה חשובה ב-ALISP. פעמים רבות אנו מעוניינים לעצור את תהליך ה-EVALUATION של ה-ALISP, ולהשתמש בערך כלשהו ללא בדיקה.

```
(SETQ X (5.000 6.0000))   -------> ERROR : BAD FUNCTION
```

קיימת כאן בעיה !!!

פעולת SETQ מבצעת : בדוק ערך שלישי בביטוי והכנס את ערכו למשתנה השני ברשימה זאת, בלי להתיחס לערכו של המשתנה השני.

במקרה שלנו (SETQ X (5.00 6.0000)), ה-ALISP בא להעריך את הרשימה (5.000 6.0000), כאשר האיבר הראשון ברשימה אינו פעולה אלא מספר ממשי.

הצורה הנכונה, לכן, היא: (SETQ X (QUOTE (5.000 6.000))

או: (SETQ X '(5.000 6.000))

ניתן היה גם לכתוב : (SETQ X (LIST 5.00 6.000))

באותה תוצאה.

ב-LIST קיימת גמישות אשר אינה קיימת ב-QUOTE, והיא זו שמאפשרת השמה של שמות משתנים.

```
( SETQ A 5.0)
( SETQ B 6.0)

( LIST A B)               --------------> (5.00000 6.00000)
(SETQ POINT1 (LIST A B))  --> (5.00000 6.00000)
```

<u>תרגיל</u>

הכנס את הערכים למשתנים הבאים:

A	6
B	3 + 2 + 5
C	A + B
PT1	קואורדינטה 1,1
PT2	קואורדינטה A,B
CITY	TEL-AVIV

בדוק את הערכים בכתיבת שם המשתנה, עם סימן קריאה.

תוספת אברים לרשימה LIST קיימת

פקודות LIST ו-QUOTE נוחות לבנית רשימות חדשות, אך פקודת CONS הינה הכלי העיקרי להוספת אברים לרשימה קיימת.

```
(LIST 2.0 3.0)

(SETQ LST (LIST 2.0 3.0))

!LST

(CONS 1.0 LST)

!LST
```

DOTTED PAIRS

DOTTED PAIR הינו סוג מיוחד של LIST.

DOTTED PAIR כולל תמיד שני אברים, המופרדים ביניהם ע"י נקודה.

דוגמאות :

```
(0  .  "LINE")

( 52  .  18)

(8  .  "0")
```

DOTTED PAIRS תופסים פחות מקום בזכרון מאשר LISTS אחרים.

תכונה נוספת של DOTTED PAIR, וחשובה לא פחות, הינה בפקודת CDR בה מוחזר ערך של ATOM ב-DOTTED PAIR בניגוד ל-LIST ברשימה אחרת.

```
(5 6)  <----      ( SETQ LLL (LIST 5 6))

(6)  <----      (CDR LLL)

(5  .  6) <----      ( SETQ DDD (CONS 5 6))

6  <----      (CDR DDD)
```

CONS יוצרת DOTTED PAIR במקרה ושני האברים ל-CONS הינם ATOMS.

השימוש העיקרי של DOTTED PAIRS הינו ברשימות הישויות בבסיס הנתונים של ה-ACAD.

<u>תרגיל</u>

הכנס למשתנה **LIST** את הערך (1.0 1.0), והוסף את האיברים הבאים לרשימה, כל אחד בנפרד:

2.0

3.0

4.0

צור זוגות **DOTTED PAIRS**

FIRST ELEMENT	SECOND ELEMENT
0	"LINE"
0	"CIRCLE"
6	"CONTINUOUS"
6	"HIDDEN"
8	"0"
8	"DIM"
62	7
70	1

זוגות אלה דומים לזוגות שנשתמש ב-**ALISP**, כדי לגשת להגדרת ישויות.

פירוק LIST לגורמים

לאחר שלמדנו לבנות LIST בעזרת פקודות LIST QUOTE ו-CONS, הגענו לכלים לטיפול בפריטים בתוך רשימה.

שליפת איבר ראשון ברשימה

משתנה PT1 מקבל ערך של קואורדינטות נקודה. מאחר וקואורדינת ה-X הינה הראשונה, דרושה פונקציה ל"שליפת" האיבר הראשון ב-LIST.

```
(SETQ PT1 (LIST 3.0 6.0))

(  3.0 6.0  ) <---- !PT1
```

פונקצית CAR מחזירה את האיבר הראשון ב-LIST.

```
3.000000  <-------        (CAR PT1  )
```

דוגמאות :

```
(1 2 3)  <--------       (LIST 1 2 3)

1  <------  (CAR (LIST 1 2 3))

((1 2 )  3)   <--------(LIST (LIST 1 2 ) 3)

(1 2)   <--(CAR (LIST (LIST 1 2 ) 3))
```

המשלים ל-CAR

המשלים ל-CAR הינו CDR.

כאשר מורידים מ-LIST את ה-CAR שלה נותר ה-CDR.

בעוד CAR יכול להיות LIST או ATOM , CDR יהיה תמיד LIST. פרט לאחד היוצא מהכלל - DOTTED PAIR.

דוגמאות:

```
        3 <--------      (CAR (LIST 3 6))
      (6) <--------      (CDR (LIST 3 6))
        1 <---------   (CAR (LIST 1 2 3))
    (2 3) <---------   (CDR (LIST 1 2 3))
  (1 2)  <---- (CAR (LIST (LIST 1 2) 3))
    (3)  <---- (CDR (LIST (LIST 1 2) 3))
NIL      () <------------- (CDR (LIST 3))
```

<u>שימוש ב-CADR, CDR, CAR ו-CADDR, ברשימות קואורדינטות</u>

פקודות **CAR CDR CADR CADDR** הן פקודות להוצאת פריטים ב-LIST.

CAR - מחזיר איבר ראשון.

CADR - " " שני.

CADDR - " " שלישי.

CDR - מחזיר רשימה, פחות איבר ראשון.

אלה מאפשרים שליפת קואורדינטות X, Y, Z מרשימה.

<u>תרגיל</u>

```
(SETQ PT1 (LIST 4. 5.))
!PT1

(CAR PT1)
(CADR PT1)
(CDR  PT1)

(SETQ BIGLIST (LIST 1. 2. 3.))

(CAR BIGLIST)
(CADR BIGLIST)
(CDR BIGLIST)
(CADDR BIGLIST)

(SETQ PT2 (CDR BIGLIST))
(SETQ X (CAR PT2))
(SETQ X (CAR (CDR BIGLIST))
(SETQ Y (CADR (CDR BIGLIST)))
(SETQ Y (CADDR BIGLIST))
```

הכנס את הערכים הבאים למשתנים:

```
PT1       (1. 2.)

PT2       (5. 6.)

LONGLIST  (3. 4. 5.)

X1        THE CAR OF PT1

X2        THE CADR OF PT1

X3        THE CAR OF PT2

Y2        THE CADR OF PT2

PT3       THE CDR OF LONGLIST
```

מתח בעזרת **ALISP-COMMAND** קו מ-PT1 ל-PT-2 ל-PT3.

כתוב ב-**ALISP** סדרת ביטויים שימצאו את נקודת האמצע בין PT1 ו-PT2, ויכניסו אותה ל-**MIDPOINT**.

שימוש בקבצי AUTOLISP

טעינת קובץ תוכנית ALISP

כל קבצי תוכניות ALISP, מסתיימים בסיומת LSP.

בהנחה כי קובץ TEST.LSP כולל תוכנית LISP, ונמצא ב-DIRECTORY העבודה שלנו, על מנת להפעילו עלינו לטעון אותו תחילה ובעזרת:

```
(LOAD "TEST")
```

הערך המוחזר הינו שם הפונקציה האחרונה שנקראה.

אם היה הקובץ ב-DIRECTORY אחר, על המשתמש היה צריך להצביע עליו, למשל :

```
(LOAD "D:/LSPDIR/TEST")
```

פרט לקובץ ACAD.LSP, הנטען באופן אוטומטי בכניסה ל-ACAD, יש לטעון כל קובץ אחר בעזרת ה-LOAD לפני ההפעלה.

יצירת קבצי תוכניות ALISP והגדרת שיגרות

קובץ תוכנית ניתן ליצירה על ידי כל **TEXT EDITOR**.

השורות הבאות יוצרות קובץ תוכנית, שתקרא **LINES.LSP**.

יש להקפיד על הדפסת האותיות הקטנות מסוג n\ ,כפי שהן מופיעות, כיוון שהן מציינות סימן מיוחד. (n\ - מציין CARRIAGE RETURN - קפיצה לשורה חדשה בזמן כתיבה).

```
(SETQ P1 (GETPOINT "\nBASE POINT : "))

(SETQ P2 (GETPOINT "\nFIRST END POINT : "))

(COMMAND "LINE" P1 P2 "")

(SETQ P2 (GETPOINT "\nSECOND END POINT :  "))

(COMMAND "LINE" P1 P2 "")

(SETQ P2 (GETPOINT "\nTHIRD END POINT :"))

(COMMAND "LINE" P1 P2 "")
```

לאחר שיצרנו ושמרנו קובץ זה על הדיסק ב-ACAD, נרשום :

```
(LOAD "LINES")
```

כדי להפעיל את הפקודות פעם נוספת, יש לבצע שוב טעינה....

יש לשים לב כי (LOAD filename), שהיא פקודת **ALISP**, ו-**LOAD** של **ACAD** מבצעות שני דברים שונים. (**LOAD** טוען **SHAPE**).

<u>תרגיל</u>

כתוב את הביטוי הבא לתוך קובץ בשם **TEST.LSP**.
טען אותו עם **LOAD**.

```
(SETQ CENPT (GETPOINT "\nCENTER POINT :"))

(SETQ RAD   (GETDIST  CENPT "\MRADIUS :"))

(COMMAND "CIRCLE" CENPT RAD)

(SETQ CENPT (LIST (CAR CENPT) (+ 1 (CADR CENPT))))

(COMMAND "CIRCLE" CENPT RAD)

(SETQ CENPT (LIST (CAR CENPT) (+ 1 (CADR CENPT))))
(COMMAND "CIRCLE" CENPT RAD)
```

הגדרת שיגרות

פקודת **ALISP - DEFUN** מגדירה שיגרה. כלומר, יוצרת פונקצית **ALISP** חדשה. זו הדרך ליצור פקודות AUTOCAD חדשות.

כאשר מפעילים פקודת **DEFUN**, מצרפים **SYMBOL** חדש לרשימת הפונקציות של **ALISP**, המכונה **ATOMLIST**.

!ATOMLIST ------> מחזיר רשימת פונקציות הALISP.

לדוגמא:

את רשימת הפקודות שיצרנו ב- LINES.LSP נהפוך לפונקציה . בדרך זו נוכל להפעיל אותה בלי לטעון אותה כל פעם כמו קודם.

```
(DEFUN LINES ()
    (SETQ P1 (GETPOINT "\nBASE POINT : "))
    (SETQ P2 (GETPOINT "\nFIRST END POINT : "))
    (COMMAND "LINE" P1 P2 "")
    (SETQ P2 (GETPOINT "\nSECOND END POINT :  "))
    (COMMAND "LINE" P1 P2 "")
    (SETQ P2 (GETPOINT "\nTHIRD END POINT :"))
    (COMMAND "LINE" P1 P2 ""))
)
```

(LOAD "LINES") ----------> LINES מחזיר

(LINES) נרשום ב-ACAD

טענו את הפונקציה מהדיסק לזכרון.

מאחר והגדרנו אותה כפונקציה, היא הפכה לאחת מפונקציות ה-**ALISP** וההפעלה תהיה כמו כל פונקציה אחרת בין סוגריים.

בפונקציה, כפי הגדרנו כאן, ניתן להשתמש במסגרת כל פונקציה אחרת של **ALISP**, כמו כל פקודה פנימית אחרת של **ALISP**.

<u>**תרגיל**</u>

הכנס הפונקציה הבאה בעזרת ה-**TEXT EDITOR** לקובץ **SQUARE.LSP**.

```
;SQUARE.LSP
;
;PROMPTS FOR LOWER LEFT CORNER AND LENGTH OF SIDE.
;DRAWS A SQUARE OUT OF FOUR LINE ENTITES.
;
;WRITTEN BY JERRY FORD
;MODIFIED BY PHIL CARTESIAN 2/2/88
;
;FOR AUTOCAD VER 2.18 AND LATER.
;
(DEFUN SQUARE ()                                    ;DEFINE SQUAR
  (SETQ LL (GETPOINT "\nLOWER LEFT CORNER :"))     ;GET LL CORN
  (SETQ SIDE (GETDIST LL "\nLENGTH OF SIDE:"))     ;GET LENGTH
  (SETQ UL  (POLAR LL (/ PI 2.0) SIDE))            ;UPPER LEFT
  (SETQ UR  (POLAR UL (* PI 2.0) SIDE))            ;UPPER RIGHT
  (SETQ LR  (POLAR UR (* 3 (/ PI 2.0)) SIDE))      ;LOWER RIGHT
  (COMMAND "LINE" LL UL UR LR "CLOSE")             ;DRAW LINES
)

(LOAD "SQUARE")

(SQUARE)
```

מספר הבהרות:

- זויות תמיד ברדיאנים.
- PI משתנה עם ערך 3.141593.
- **POLAR** מוצא קואורדינטת נקודה, לפי מרחק וזוית מנקודה נתונה.

<u>פונקציות הדורשות ARGUMENTS</u>

הפונקציות הקודמות שיצרנו LINES ו-SQUARE הינן פונקציות שאינן דורשות פרמטרים ARGUMENTS. להפעלתן יש להקיש את שמן בסוגריים ותו לא.
(LINES) או (SQUARE).

שתי הפונקציות הבאות הן דוגמאות לפונקציות הדורשות פרמטר אחד.

במילים אחרות - הפונקציות דורשות ערך או משתנה לפעול עליו.

ALISP משתמשת בהגדרת זויות ברדיאנים. אנו, לעומת זאת, רגילים למדוד זויות במעלות. לכן, פונקציה שתתרגם מעלות לרדיאנים ולהיפך יכולה להיות שימושית.

שיגרת DTR מבצעת הפיכה של מעלות לרדיאנים:

```
;  DTR.LSP
;
; THIS IS A LISP UTILITY FOR CONVRTING DEGREES TO RADIAN.
; THIS FUNCTION HAS ONE ARGUMENT D WHICH IS THE NUMBER OF
; DEGREES YOU WANT TO CONVERT TO RADIANS.
; EXAMPLE (DTR 180) WILL RETURN NUMBER OF RADIANS IN 180
; DEGREES.
;
; WRITTEN BY KELVIN TROOP. FOR USE WITH ACAD 2.18 AND LATER.
;
(DEFUN DTR  (D)             ;DEFINE DTR AND GET DEGREES

   (/ (* D PI ) 180)        ; CALCULATE RADIANS
)
```

שיגרת RTD מבצעת הפיכה של רדיאנים למעלות :

```
;  RTD.LSP
;
;
; THIS IS A LISP UTILITY FOR CONVRTING RADIANS TO DEGREES
; THIS FUNCTION HAS ONE ARGUMENT R WHICH IS THE NUMBER OF
; RADIANS YOU WANT TO CONVERT TO DEGREES.
; EXAMPLE   (RTD PI) WILL RETURN THE NUMBER OF DEGREES
; IN PI RADIANS.
;
; WRITTEN BY KELVIN TROOP. FOR USE WITH ACAD 2.18 AND LATER.
;
(DEFUN RTD  (R)             ;DEFINE RTD AND GET RADIANS
   (/ (* 180 R) PI)         ; CALCULATE DEGREES
```

לאחר הטענת הקבצים:

```
(LOAD "DTR")
(LOAD "RTD")
```

נפעיל אותן:

```
3.141593<-----    (DTR 180)

180.0000<------    (RTD PI)
```

<u>הפעלת פקודת ALISP באמצע פקודת AUTOCAD</u>

למרות שרוב שיגרות ה-ALISP מבוצעות מרמת ה-COMMAND ב-AUTOCAD, הרי לפעמים נוח להפעיל שיגרה באמצע פקודה.

למשל תוך פקודת LINE , ACAD מצפה לנקודה אך הנקודה הדרושה הינה יחסית ל-ENDPOINT של קו קיים. ה-AUTOCAD מאפשר התייחסות רק יחסית לנקודה אחרונה. במקרה שלנו, צריך להצביע על נקודה שאינה אחרונה.

השיגרה בתרגיל הבא הקרויה REF.LSP, פותרת בעיה זו בצורה אלגנטית.

REF מבקש נקודה, מכניס את ערכה למשתנה מערכת LASTPOINT ואז ניתן להצביע על המיקום היחסי בכל שיטה מקובלת, כולל יחסי ופולרי.

<u>תרגיל:</u>

```
;REF.LSP
;
;THIS FUNCTION ALLOWS THE USER TO SPECIFY A POINT REFERENCED
;FROM ANOTHER POINT WHILE IN THE IDDLE OF A COMMAND
;
;WRITTEN BY DUFF KURLAND
;
;FOR AUTOCAD 2.5 AND LATER

(DEFUN REF ()
  (SETVAR "LASTPOINT" (GETPOINT "\nREFERENCE POINT : "))
  (GETPOINT (GETVAR "LASTPOINT") "\nTO POINT :"))
```

הטען ב-LOAD והפעל:

```
LINE
from point 1,1
to point (REF)
reference point: 4,4
to point :@1,0
to point : <RETURN>
```

הגדרת בעיה ובניית התוכנה

אנו משתמשים ב-ALISP מאותן סיבות שמשתמשים בכל שפת תכנות - ככלי לפתרון בעיות.

ננסה להתחקות על מצב בעייתי ב-AUTOCAD, ולראות כיצד ניתן לפתור אותו.

בעיה: מספר קוים בשרטוט מכני שורטטו בעט לא נכון , זאת משום שהקוים קבלו מאפיין COLOR שונה מה-LAYER שלהם.

פתרון: יש לדאוג לכך שהקוים הנ"ל מקבלים COLOR על-פי ה-BYLAYER.

דרך: א. פתרון בפקודות AUTOCAD.

ב. פתרון פקודות MENU.

ג. פתרון בתכנות ALISP.

א. **פתרון בפקודות AUTOCAD**

פירוט הצעדים לפתרון: הפוך את כל הקוים ל-COLOR BYLAYER.

* THAW (הפשר) כל השכבות.

* הדלק (ON) את כולם.

* ZOOM ALL, כדי לראות את כל השרטוט.

* CHANGE עם SELECT, לכל הנמצא בשרטוט בעזרת CROSSING.

* המשך CHANGE עם PROPERTIES - COLOR ו-BYLAYER.

```
COMMAND: LAYER THAW  * ON *
COMMAND: ZOOM A
COMMAND: CHANGE C שתי נקודות בקצוות המסך
         PROPERTIES  COLOR BYLAYER
```

את רצף הפקודות הזה נצטרך להקיש בכל פעם שנפגוש בעיה זו.

זהו רצף ארוך לפעולה פשוטה ושכיחה, לכן מומלץ להופכו לרצף MACRO.

ב. <u>פתרון ברמת MENU MACRO</u>

אותו רצף ניתן לפתור בפקודת MACRO:

```
[CBYLR]^C^CLAYER;T;*;ON;*;;ZOOM;A;CHANGE;C;\\;P;C;BYLAYER;;
```

אחרי בחירה מהתפריט של **CBYLR**, התהליך מתבצע אוטומטית עד לציון שתי נקודות במסך לפקודת ה-**CHANGE**.

ג. <u>פתרון ברמת תכנות</u>

ישנם מספר חסרונות בפתרון התפריט שרשמנו:

א- יש למצוא את דף התפריט בו כתובה הפקודה **CBYLR**.
ב- יש לציין שתי נקודות במסך.
ג. יש כאן פוטנציאל להמתנה לשני **REGEN** (אחד ב-**ZOOM** ואחד ב-**THAW**).

בשימוש ב-**ALISP** ניתן לעקוף את שלושת החסרונות הנ"ל.

שלב ה-**THAW** ושלב ה-**ZOOM ALL** מיותרים מבחינה פונקציונלית.

```
(COMMAND "CHANGE" (SSGET "X") "" "P" "C" "BYLAYER"  "")
```

נותר רק ליטוש אחרון, והוא להפוך זאת לשיגרה.

```
(  DEFUN  C:CBYLR    ()
(COMMAND "CHANGE" (SSGET "X") "" "P" "C" "BYLAYER"  ""))
)
```

הפעלה ע"י **CBYLR**.

שיקולים בעבודה בתכנות

1. הגדר הבעיה בשפה פשוטה.
2. פרט שלבים בפתרון אפשרי.
3. חלק כל שלב לשלבים הבסיסיים ביותר.
4. לבדוק האם הפתרון אפשרי ברמת ה-**COMMAND**.
5. הערך האם הפתרון כדאי לאוטומציה?
   - מבחינת השקעת הזמן בכתיבת **MACRO**.
   - מבחינת תיעוד והוראה למשתמשים.
   - הערכת הרווח הצפוי בהגדלת כושר היצור במערכת.
6. בחן האם הפתרון דורש יכולת, שאינה קיימת ב-**MACRO**.
   - חישובים אריטמטיים.
   - איכסון וקריאה לערכים במשתנים.
   - בדיקה לתנאים.
   - חזרה על פקודות או איטרציות.
7. כתוב את רצף פקודות ה-**ALISP**.
8. בצע בקרת איכות על-ידי הפעלה בתנאים שונים.
9. הכן תיעוד והדרכה למשתמשים.

יצירת פקודות AUTOCAD חדשות

ניתן ליצור פקודות AUTOCAD חדשות בעזרת **DEFUN**, כך שיתפקדו כפקודות **AUTOCAD**.

שם הפקודות ב-**DEFUN** חייב להיות בצורה **C:NAME**, כאשר כל האותיות הן גדולות (CAPITAL LETTERS). יש להקפיד לא להשתמש בשמות פקודות **AUTOCAD** קיימות. הפקודה צריכה להיות מוגדרת בלי רשימת פרמטרים.

לדוגמא, הגדרת שם מקוצר ל REDRAW:

```
(DEFUN C:R () (COMMAND "REDRAW"))
```

ההפעלה מבוצעת ע"י הקשת R.

דוגמא נוספת: פקודת **AUTOCAD** חדשה למחיקת כל הנמצא בתוך ה-**LIMITS**.

```
(DEFUN C:ERASELIM ()
  (SETQ MIN (GETVAR "LIMMIN"))
  (SETQ MAX (GETVAR "LIMMAX"))
  (COMMAND "ERASE" "C" MIN MAX "")
  (COMMAND "REDRAW"))
```

הפעלה ע"י הקשת **ERASELIM**.

<u>תרגיל</u>:

הגדר הפונקציות הבאות, כפקודות AUTOCAD חדשות:

```
ZW       (DEFUN C:ZW () (COMMAND "ZOOM" "W"))
ZP       (DEFUN C:ZP () (COMMAND "ZOOM" "P"))
ZA       (DEFUN C:ZA () (COMMAND "ZOOM" "A"))
LINE1    (DEFUN C:LINE1 () (COMMAND "LINE" PAUSE PAUSE ""))
C1       (DEFUN C:C1 () (COMMAND "CIRCLE" PAUSE 1.0))
DELAL    (DEFUN C:DELAL () (CAMMAND "ERASE" (SSGET "X") ""))
R90      (DEFUN C:R90 () (SETQ SSET (SSGET)) (COMMAND
           "ROTATE"  SSET "" PAUSE 90.0))
```

משפטי התניה ופעולות לוגיות

ערכי ביטויים

ה-ALISP לוקחת משפט קלט של התוכנית, מעריכה אותו ומחזירה ערך תוצאה. ALISP תמיד מחזירה תוצאה במבנה אחד מסוגי האינפורמציה החוקיים. כלומר: LIST ,TEXT , STRING , INTEGER , REAL. ולפעמים תהיה התוצאה בצורת T או NIL.

T ו-NIL

T הוא המקביל ב-ALISP ל"נכון" או "אמת", ובהתאמה NIL יהיה מתאים ל "שקרי" או "לא נכון" .

ביטוי ALISP שערכו מתאים ל-T אינו יכול להיות NIL ולהיפך.

ALISP כולל מספר פקודות הבודקות לתנאים שונים ומחזירות T או NIL. הפקודות מתחלקות לשתי קבוצות : PREDICATE ו-LOGICAL.

PREDICATES למספרים

PREDICATES מחזירים תמיד T או NIL. הם משווים תכונה בשני ביטויים.

דוגמא : שימוש ב-PREDICATE =

```
( =  4.0 (+ 2.0 2.0 ))
( = 4.0 (+ 1.5 2.0 ))
```

יחזיר T אם הם שווים בינהם ו-NIL אם הם שונים.

PREDICATE מסוג > בודק אם מספר אחד גדול מהשני.

```
(< 45.125 (- 150.00 100.00))
```

אותה פעולה ניתנת לביצוע בין ערכי משתנים.

```
(SETQ NUM1 15.75)
(SETQ NUM2 7.00)
```

הביטוי הבא בודק אם ערך משתנה NUM1 גדול או שווה ל-NUM2:

```
(>= NUM1 NUM2)
```

רשימת PREDICATE למספרים.

=	שווה
/=	לא שווה
<	פחות
<=	פחות או שווה
>	גדול
>=	גדול או שווה
MINUSP	פחות מאפס
ZEROP	שווה לאפס

<u>תרגיל</u>

כתוב ביטוים הבודקים את התנאים הבאים:

בדיקה	שני	ראשון
EQUAL	1.0	1
LESS THAN	2.0	1.0
LESS OR EQUAL TO	100/3	33
LESS OR EQUAL TO	100/3.0	33
LESS THAN ZERO		7
EQUAL TO ZERO		0.000001

הכנס למשתנים הבאים את הערכים, ובדוק עבור: קטן, שווה וגדול.

ערך	משתנה
10.0	LOW
50.0	HIGH

PREDICATES נוספים

פקודות נוספות לטיפול במשתנים:

ATOM	האם המשתנה הינו ATOM.
LISTP	האם המשתנה הינו LIST.
BOUNDP	האם יש ערך במשתנה.
NULL	האם המשתנה שווה ל-NIL.
NUMBERP	האם המשתנה הינו מספר.
EQUAL	האם האברים שווים.
EQ	האם האברים שווים לאותם ערכים.

דוגמא:

```
(SETQ A 'UNIT)
(SETQ L '(+ 1 2 ))
(SETQ N 5.0)

(ATOM A)      -->  T
(ATOM L)      -->  NIL
(ATOM N)      -->  T
```

שימוש בפונקצית TYPE

ניתן בעזרת פונקצית **TYPE** לבדוק כל משתנה, לסוג הנתונים בתוכו.

```
(SETQ X 4.5)    --> 4.5
(EQUAL (TYPE X) 'REAL)    --> T
(EQUAL (TYPE X) 'INT )    --> NIL
```

פקודות לוגיות

NOT - הפקודה מחזירה ערך הפוך לערך הנבדק.

```
(NOT NIL)   ---> T

( NOT T)    ---> NIL
```

AND - הפקודה בודקת איבר אחד, או יותר, אם יש בינהם **NIL** ומחזירה **NIL**. בכל מקרה אחר מחזירה T.

דוגמא:

```
(SETQ  PT1 '(1.0 2.0) PT2  '(2.0 2.0) PT3 '(4.0 3.0)    )

(AND PT1 PT2 PT3)    ------>  T

(SETQ PT3 '() )

(AND PT1 PT2 PT3)    ------> NIL
```

OR - הפקודה בודקת איבר אחד, או יותר, אם יש בינהם T ומחזירה T. בכל מקרה אחר מחזירה **NIL**.

דוגמא:

```
(SETQ  PT1 '(1.0 2.0) PT2  '(2.0 2.0) PT3 '(4.0 3.0)    )

(OR PT1 PT2 PT3)    ------>  T

(SETQ PT3 '() )

(OR PT1 PT2 PT3)    ------> T

(SETQ PT2 '() )

(OR PT1 PT2 PT3)     -----> T

(SETQ PT1 '() )

(OR PT1 PT2 PT3)     -----> NIL
```

משפטי בדיקה שימושיים בפקודות תנאי **IF** או לולאות **WHILE**, בעזרתם מבקרים את זרימת התוכנית. בדרך זו ניתן להורות לתוכנה לבצע פעולות שונות, כתוצאה מקיום מצבים שונים.

IF

IF - פקודת **IF** מופיעה בכל שפת תכנות.

הפקודה שואלת "האם זה נכון ש.... ?" וכאן בא התנאי הנבדק.

ב-ALISP השאלה הינה בסגנון "האם זה T ?"

מבנה הפקודה:

(IF תנאי <משפט אם כן> <משפט אם לא>)

לדוגמא:

```
(SETQ PT1 (LIST 1.0 1.0)
      PT2 (LIST 2.0 2.0)
)

(IF (AND (LISTP PT1) (LISTP PT2) )
    (COMMAND "LINE" PT1 PT2 "")
    (PROMPT "\nNO ARGUMENTS ")
)
```

בדוגמא הקודמת קיימת שגיאה קטנה , בגלל הערך הכפול של NIL (גם LIST וגם ATOM). גם אם יש NIL ב-PT1 ו-PT2, יבוצע הקו. לכן דרושה עוד בדיקה ל-NULL.

דוגמא :

```
(SETQ SM  (GETVAR "SNAPMODE")  )

(IF (= SM 0)
    (PRINT "\nSNAP IS OFF")
    (PRINT "\nSNAP IS ON ")
)
```

פקודת ה-**IF** אינה מאפשרת ביצוע של יותר מפעולה אחת עבור משפט ה"אם התנאי מתקיים" וכן גם עבור ה"אם לא".

על קושי זה מתגברים בעזרת פונקצית ה-**PROGN**. פונקציה זו משמשת להערכה של מספר ביטויים, כאשר רק אחד אפשרי.

```
(SETQ PT1 (LIST 1.0 1.0)
      PT2 (LIST 3.0 3.0)
)

(IF (AND (LISTP PT1) (LISTP PT2) )
    (PROGN (COMMAND "LINE" PT1 PT2 "")
           (PROMPT "\nCREATED ONE LINE THANK YOU.")
    )
)
```

<u>תרגיל:</u>

טען המשתנים בערכים :

(1.0 1.0)	PT1
(5.0 5.0)	PT2
(7.0 5.0)	PT3

רשום ביטויים:

- האם PT3 שווה לNIL.
- אם PT3 הוא NIL, מותח קו מ-PT1 ל-PT2.
- אם PT3 הוא נקודה, מותח קשת לפי 3 נקודות. על פני שלוש הנקודות.

  נסה את הביטוי, כאשר PT3 הוא NIL.

לולאות וחזרה על סדרות פקודות

החזרה על הפקודות והלולאות הן בין המאפיינים החשובים של שפות תכנות. אם, לדוגמא, ישנה קבוצת פקודות הרצויה לביצוע חמש פעמים, אין היגיון רב בכתיבה של חמש פעמים רצופות של אותה סדרה.השימוש בפקודת לולאה יקצר את התהליך.

פקודת - REPEAT

```
(REPEAT 3  (PROMPT "\nHELLO" ) )
```

ה-**NIL** בסוף נובע מכך שערך פעולת (PROMPT) הוא **NIL**.

כדוגמה, נבנה פונקציה פשוטה הסופרת:

הפונקציה מדפיסה את ערך התא לתוכו סופרים, ואחר מוסיפה 2 לתא.

```
(DEFUN COUNT ()
   (SETQ NUM 0 )
   (REPEAT 10
       (PRINT NUM)
       (SETQ NUM (+ NUM 2)
   )
)
```

דוגמא מעשית יותר, תהיה פונקציה שתמחוק את 10 הישויות האחרונות שהוכנסו לשרטוט.

```
(DEFUN ERASE10 ()
   (REPEAT 10
       (COMMAND "ERASE" "L" "")
   )
)
```

פונקציה זו נפעיל, כאשר יש בשרטוט לפחות 10 ישויות.

<u>תרגיל</u>

```
(SETQ LL (LIST 0.0 0.0)
(SETQ UR (LIST 12.0 9.0)
(COMMAND "ZOOM" "W" LL UR)

(SETQ CIRCEN (LIST 5.0 5.0))
(COMMAND "CIRCLE" CIRCEN 1.0)

(SETQ RTDSPL (LIST 0.5 0.0)
(SETQ LFDSPL (LIST -0.5 0.0))
(SETQ UPDSPL (LIST 0.0 0.25))
(SETQ DNDSPL (LIST 0.0 -0.25))

(REPEAT 10 (COMMAND "MOVE" "L" "" RTDSPL ""))
(REPEAT 5 (COMMAND "MOVE" "L" "" UPDSPL ""))
(REPEAT 7 (COMMAND "MOVE" "L" "" LFDSPL ""))
(REPEAT 3 (COMMAND "MOVE" "L" "" DNDSPL ""))
```

לולאת WHILE

פונקצית ה-REPEAT הינה כלי מעולה ללולאות, שמספר החזרות של אותה סדרה ידוע. אך ברוב המקרים אין הדבר כך. במקרים שמספר ההחזרות אינו ידוע, הלולאה היא על תנאי .

במקרים כאלה, קיימת פונקצית ה-WHILE. כל זמן שהתנאי מתקיים הלולאה מתבצעת.

מבנה הפקודה הוא :

```
(WHILE    <T משהו הוא>
      (בצע את א)
      (בצע את ב)
       .
       .
       .
  <בדוק אם התנאי עדיין מתקיים
   אם כן חזור ללולאה
   אם לא צא מהלולאה   >
```

מאחר והדרך היחידה לצאת מלולאת WHILE היא בשינוי מצב התנאי, מומלץ לכן לכלול בתוך הלולאה פקודה המשנה את ערך המשתנה לבדיקה.

לדוגמא: נכתוב תוכנית המזיזה את הישות האחרונה שהוכנסה, ביחידה אחת לכוון ה-Y. זאת כל זמן שהמשתמש בתוכנית מאשר בהקשת Y.

לפני כן, נגדיר בפירוט דרישות מהתוכנית:

- יש להגדיר פונקציה.
- יש להכניס למשתנה את ערך התזוזה הדרושה.
- יש לבקש אישור המפעיל לכל פעולת הזזה.
- כל זמן שהתשובה היא כן, יש לבצע. כל תשובה אחרת תגרום ליציאה מהלולאה.
- שמות משתנים קצרים ובעלי משמעות.
- כתיבה ברורה לקריאה קלה - הערות וסידור סוגריים .
- תיעוד, כולל: מי ומתי נכתבה התוכנית, ושינויים מתי, על ידי מי.
- גירסת AUTOCAD, עבורה נכתבה התוכנית.

```
;
;  MOVEUP.LSP
;
;  THE FUNCTION MOVEUP PROMPTS THE OPERATOR WHETHER TO MOVE
;  ENTITY IN DATABASE THAT IS VISIBLE ON THE DISPLAY
;  ONE UNIT IN THE Y DIRECTION.
;

;  WRITTEN BY PHIL CARTESIAN  2/5/88
;

;  FOR AUTOCAD VERSION 2.5 AND LATER.
(DEFUN MOVEUP ()
  (SETQ DISPLC (LIST (0.0 1.0))
  (SETQ OPTEST
      (GETSTRING "\nMOVE LAST UP 1 UNIT (Y/N) <N> : ")
  )
  (WHILE
     (OR
        (EQUAL OPTEST "Y")
        (EQUAL OPTEST "y")
     )
     (COMMAND "MOVE" "L" "" DISPLC "")
     (SETQ OPTEST
        (GETSTRING "\nMOVE LAST UP 1 UNIT (Y/N) <N> :")
     )
   )
)
```

<u>תרגיל</u>

* הדפס את השורות הבאות:

```
(setq count 0)
(while (< count 5) (print count) (setq count (1 + count (1 + count)))

(setq test "Y")

(while
  (equal test "Y")
  (prompt "\nYour answer is yes.")
  (setq test (getstring "\nAgain? :"))
)
```

Create the file CFIVE.LSP. LOAD the file into the current Drawing Editor session, and use the function CFIVE.

```
; Cfive.lsp.
;
; Function CFIVE prompts the operator for the center point
; and radius of a circle. If the radius of the circle is less
; than 1 unit, the function prompts the operator to choose
; another radius; otherwise, 5 circles are drawn with the specified
; radius, equally spaced at a user-specified distance
; and direction.
;
; Written by phil 2/5/88.
;
; For AutoCad version 2.5 or later.
```

```
(defun cfive ()
;define function
  (setq count 0)                                        ;zero
out count
  (setq cenpt                                           ;get
center point
    (getpoint "\nCenter of first circle: ")
)
(setq rad (getdist cenpt "\nRadius: "))                 ;get
radius
(while
  (< rad 1.0)
radius too small?
  (prompt "\nRadius is too small <less than 1>. ")
  (setq rad
    (getdist cenpt "\nEnter a larger radius: ")
  )
)
(setq dispt (getpoint cenpt "\nDisplacement: "))  ;get to-point
(setq disang (angle cenpt dispt)                    ;angle
for copies
      disdst (distance cenpt dispt)                 ;distance
for copies
)
(while
  (< count 5)
;drawn 5 circles?
  (command "circle" cenpt rad)                      ;draw
circle
  (setq cenpt (polar cenpt disang disdst));change center point
  (setq count (1+ count))
;increase count
 )
)
```

SELECTION SETS

שימוש ב- SELECTION SET ב- ALISP, דומה מאד לשימוש ב-SELECT ב-AUTOCAD. כאשר מקבלים את ההודעה SELECT OBJECTS, בוחרים בעזרת חלון אחרון או ישות - ישות בנפרד, ואז מפעילים פעולה על כל הישויות שנבחרו.

זו העת ליצור מספר ישויות בשרטוט, כדי להשתמש בהן בהמשך.

שרטט מס' מעגלים וקווים על המסך:

```
<SELECTION SET : 1> <------     (SETQ S1 (SSGET))
```

כלומר, נוצר SELECTION SET והוכנסו לתוכו הישויות הנבחרות:

```
LIST
!S1
```

מקבלים רשימה של ישויות בתוך S1. כך ניתן "לצבור" ישויות בתוך SELECTION SET, ולהפעיל עליהן פעולות שונות במהלך השרטוט.

SSGET FILTERS

ALISP מאפשרת סינון של בחירת ישויות, לפי מאפיינים שונים.

לדוגמא: אם נרצה לבחור SELECTION SET, שיכלול את כל הקוים LINE, נרשום:

```
(SETQ S1 (SSGET "X" (LIST (CONS 0 "LINE"))))
(SETQ S1 (SSGET "X" '((0 . "LINE")))
```

ה- X נקרא פילטר, והוא מציין שיש לבדוק את כל הישויות מולו. ישות מתאימה מוכנסת ל-SELECTION SET.

אם אנו מעונינים בכל העיגולים אשר בשכבה בשם HID, נרשום:

```
(SETQ S1 (SSGET "X"  '((0 . "CIRCLE") (8 . "HID"))))
```

או:

```
(SETQ S1 (SSGET "X" (LIST ((CONS 0 "CIRCLE") (CONS 8 "HID"))))
```

<u>מספר הישויות ב-SELECTION SET</u>

פונקצית SSLENGTH מחזירה את מספר האברים ב-SELECTION SET:

```
(SETQ SLEN (SSLENGTH S1))
```

<u>קבלת שם הישות הראשונה ב- SELECTION SET</u>

```
(SETQ EFIRST (SSNAME S1 0)
```

המספר בפקודת SSNAME מציין את מספר האיבר. למשל, איבר חמישי נקבל ע"י (SSNAME S1 4):

<u>הוספת ישויות ל-SELECTION SET קיים</u>

בהנחה כי קיים כבר S1, הכולל מספר ישויות, נצייר קו ונוסיף אותו ל-S1 על ידי כתיבת :

```
(SETQ S1 (SSADD (ENTLAST) S1))
```

מוסיפים את הקו האחרון ל-S1 ומכניסים אותם ל-S1.

גישה לישויות

שמות ישויות

לכל ישות בשרטוט AUTOCAD יש שם עבור AUTOLISP. מספר טעמים לכך:
- כדי **למצוא** ישות או קבוצת ישויות.
- כדי **להפעיל** פקודות AUTOCAD עליהן.
- כדי **לבדוק** מאפייני ישות.
- כדי **לשנות** מאפיינים.

שמות הישויות טובים רק לאותה ישיבה מול ה-AUTOCAD. בכל כניסה ל-ACAD מקבלות כל הישויות שמות חדשים.

ישויות חדשות יכולות להווצר רק על ידי פקודות AUTOCAD. כמובן אין זה מכשול, משום שכל פקודות AUTOCAD יכולות להיות מופעלות בפונקציות AUTOLISP.

מציאת שם ישות

הבה ניצור קו LINE מ-1,1 ל-5,5 בשכבה 0.

```
(SETQ ENTNAME (ENTLAST)) ---> <ENTITY NAME : 60000013C>
```

ENTLAST הינה פונקציה, המחזירה שם של הישות שהוכנסה אחרונה ל-AUTOCAD.

מה ניתן לעשות עם **ENTAME** ?

למחוק את הישות. למשל: (COMMAND "ERASE" ENTNAME "")

בצורה כזו, מוחקים כמו בפקודת AUTOCAD ERASE עם אופציה L. בהבדל אחד - שם הישות נמחקת, בתנאי שהיא מוצגת על המסך.

אפשרות נוספת לבחור בישות היא באמצעות הצבעה בעכבר:

```
(SETQ ENTNAME (CAR  (ENTSEL)))
```

ENTSEL מחזיר יותר מידע מאשר רק שם הישות. אך אנו, בעזרת **CAR**, שולפים את הדרוש לנו. למעשה, מוחזרים שם הישות וקואורדינטת X Y של נקודת ה-**PICK**.

הדוגמא הבאה, היא פונקציה המבצעת העתקה : SICOPY -- SINGLE COPY

```
(DEFUN C:SICOPY ()
   (SETQ ENTNAME
         (CAR (ENTSEL) )
   )
   (COMMAND "COPY" ENTNAME "" PAUSE PAUSE ))
)
```

תרגיל

- כתוב ביטוי:
1. המזיז MOVE את הישות האחרונה בבסיס הנתונים של ה-ACAD.
2. מבקש מהמשתמש נקודת בסיס ונקודה נוספת.

- כתוב קבוצת ביטויים שיבצעו:
1. מבקש מהמשתמש לבחור ישות.
2. משנה צבע ישות לאדום.

- כתוב פונקציה שתשנה LAYER של ישות שהמשתמש בחר.
1. מבקש מהמשתמש לבחור ישות.
2. משנה את ה- LAYER של הישות.
3. מבקש מהמשתמש את השם החדש של הישות.

<u>רשימות מאפייני ישות.</u>

הגיע הזמן לבחון את המידע האגור בבסיס הנתונים של ה- ACAD,תחת שם הישות.

תחילה נצייר קו מ- 1,1 ל-5,5 בשכבה 0 ובעל התכונות הבאות:

```
COLOR           BYLAYER
LINETYPE        BYLAYER
ELEVATION       0
THICKNESS       0
```

לאחר מכן נכתוב:

```
(SETQ ENAME (ENTLAST))
```

או:

```
(SETQ ENAME (CAR (ENTSEL)))

!ENAME
```

זה אמור להחזיר את שם הישות:

```
(SETQ ELIST (ENTGET ENAME))
```

כאן מקבלים רשימה ארוכה שאם נכתוב אותה בצורה מסודרת:

```
( (-1 . <ENTITY NAME : 6000000153> )
  (0 . "LINE")
  (8 . "0")
  (10 1.000 1.000)
  (11 5.000 5.000)
)
```

הרשימות מסודרות על פי הקודים הבאים:

שם הישות	-1
סוג	0
שכבה	8
נק. התחלה	10
נק. סיום	11

פונקצית ASSOC מאפשרת שאילתות פשוטות ברשימה זו:

(8 . "0") <--------------- (ASSOC 8 ELIST)

"0" שם השכבה. <-------------- (CDR (ASSOC 8 ELIST))

<u>תרגיל</u>

- שרטט בשכבה 0,מעגל ברדיוס 1.5 ומרכז ב- 5.5,4.75.
- הדפס את הביטויים הבאים:

ביטוי	תגובת ה- ALISP

```
(setq ename (entlast))                  Entity name: 6000000028>
!ename                                  Entity name: 6000000028>

(setq elist (entget ename))           (
                                        (-1 . <Entity name: 6000000028>)
                                        (0 . "CIRCLE")
                                        (8 . "0)
                                        (10 5.500000 4.750000)
                                        (40 1.500000)
                                      )

(assoc -1 elist)                        <Entity name:6000000028>)
(assoc 0 elist)                         (0 . "CIRCLE")
(cdr (assoc 0 elist))                   "CIRCLE"
(setq etype (cdr (assoc 0 elist)))      "CIRCLE"
!etype                                  "CIRCLE"

(assoc 10 elist)                        (10 5.500000 4.750000)
(cdr (assoc 10 elist))                  (5.500000 4.750000)
(setq cpt (cdr (assoc 10 elist)))       (5.500000 4.750000)
!Cpt                                    (5.500000 4.750000)

(command "circle" cpt 3.0)
(command "circle" '(3.0 3.0) 0.5)

(command "line" cpt (cdr (assoc 10 (entget (entlast)))) "")
```

<u>מעבר על בסיס הנתונים</u>

לקראת הקטע הבא, מומלץ להכין מספר ישויות על המסך.

ENTLAST מחזיר את הישות האחרונה שהוגדרה.

ENTSEL מחזיר ישות שבחרנו.

נוח לעבור בצורה שיטתית על הישויות כולן, מהראשונה עד האחרונה בהן.

הדבר נעשה בפקודת **ENTNEXT**, כאשר היא מצויה בתוך לולאת **WHILE**.

(SETQ ENAME (ENTNEXT)) מחזיר את שם הישות הראשונה .

במקרה שאין ישות, הוא מחזיר NIL.

(SETQ ENAME2 (ENTNEXT ENAME)) - מחזיר את הישות הבאה .

הארגומנט בפקודת ה- **ENTNEXT** מורה לפקודה איזו ישות הבאה להחזיר.

הבה נרשום תוכנית, שתעבור על בסיס הנתונים ותחזיר את שמות הישויות וסוגיהן.

הדגשים:

- שמות משתנים קצרים ובעלי מובן.
- הדגשת רמות הסוגריים ע"י כניסה.
- הכנסת הערות.
- תיעוד: מה עושה התוכנית מי כתב, מתי ולאיזו גירסת ACAD.

```
; EPAGE.LSP
;
; WRITTEN BY PHIL CARTESIAN 1/28/88
;
; THIS FUNCTION PRINTS THE NAME OF EVERY ENTITY IN
; A DRAWING AND IT'S ENTIY TYPE IN THE COMMAND  PROMPT AREA.
;
; FOR USE WITH VERSION 2.5 OR LATER.
;
 (DEFUN EPAGE ()
    (SETQ ENAME (ENTNEXT))         ;GET FIRST ENT NAME
    (WHILE
       (NOT (NULL ENAME))          :OUT OF ENTITIE   ?
       (PRINT ENAME))              ;PRINT NAME
       (PRINT                      ;PRINT TYPE
          (CDR (ASSOC 0 (ENTGET ENAME)))
       )
       (TERPRI)                    ;PRINT BLNAK LINE
       (SETQ ENAME (ENTNEXT ENAME))  ;GET NEXT ENTITY
    )
    (PROMPT "\nPAGING IS FINISHED.") ;OPERATOR END MESSAGE
    (PRIN1)                          ;SUPPRESS NIL
 )
```

<u>תרגיל</u>

צייר שלוש ישויות והדפס את הביטויים:

```
(SETQ ENAME (ENTNEXT))
(COMMAND "LIST" ENAME "")
(SETQ ENAME2 (ENTNEXT ENAME)
(COMMAND "LIST" ENAME2 "")
(COMMAND "MOVE" ENAME "" PAUSE PAUSE )
(COMMAND "ERASE" ENAME2 "")
(SETQ ENAME2 (ENTNEXT ENAME))
(COMMAND "LIST" ENAME "")
(COMMAND "LIST" ENAME2 "")
```

שנה את EPAGE.LSP, הנ"ל על מנת להדפיס באותו LAYER שהישויות נמצאות בו, בנוסף לשם הישות והסוג שלה.

שינוי תכונות ישות

למדנו להשתמש ב-**ASSOC**, כדי לדעת לחפש בין המאפיינים השונים של ישויות בשרטוט .

ראינו את **IF**, ע"מ לבדוק באם המאפיינים עונים על תנאים שונים.וראינו גם כיצד ניתן לבצע לולאות - ב **WHILE**,כדי לחזור על אותה פעולה עבור מספר רב של ישויות.

בהנחה כי ישות עונה על מאפייני החיפוש, אנו יכולים לשנות אותה ישירות בבסיס הנתונים,בעזרת פקודות **SUBST** ו- **ENTMOD**.

SUBST מחליף ערך ב-**LIST** בערך אחר.

ENTMOD משנה ישירות ערך ישות, כאשר הוא מורה ל-AUTOCAD לעדכן ישות, בהתבסס על שינויים שבוצעו ברשימת המאפיינים של הישות.

הפקודות הבאות, יוצרות **LIST** במשתנה **TESTLIST** ומחליף אותו לערך חדש:

```
(SETQ TESTLIST (LIST "HIDDEN" "CENTER" "DIM"))

!TESTLIST

(SUBST "DASHED" "HIDDEN" TESTLIST)

!TESTLIST

(SETQ TESTLIST (SUBST "DASHED" "HIDDEN" TESTLIST))

!TESTLIST
```

עבודה ברשימת מאפיינים אינה שונה מהעובדה בדוגמא הקודמת:

נשרטט קו מ- 1,1 ל- 5,5, בשכבה 0.

```
(SETQ ENAME (ENTLAST))
(SETQ ELIST (ENTGET ENAME))
(ASSOC 10 ELIST)  ------> (10 1.000 1.000)
(SETQ OLD (ASSOC 10 ELIST ))
(CONS 10 (LIST 6.0 2.0))
(SETQ NEW (CONS 10 (LIST 6.0 2.0)))
(SUBST NEW OLD ELIST)
!ELIST
(SETQ ELIST (SUBST NEW OLD ELIST))
!ELIST
```

שינינו את הערך של נקודת התחלת הקו, בתוך **ELIST**, מ- 1,1 ל- 6,2 .

בבסיס הנתונים לא שינינו דבר , השינוי עצמו מתבצע בפקודת:

```
(ENTMOD ELIST)
```

ENTMOD אינו יכול לשנות את סוג הישות. למשל מ-LINE ל-CIRCLE.

פונקצית CLZERO עוברת על בסיס הנתונים,מוצאת שם כל קו בשכבה שאינה 0, ומשנה אותו לשכבה 0.

```
;  CLZERO.LSP
;
;  WRITTEN BY PHIL CARTESIAN 1/30/88
;
;  THIS FUNCTION PLACES EVERY LINE ENTITY IN A DRAWING
;  ONTO LAYER 0.
;
(DEFUN CLZERO (/ ENAME ELIST) ;DEFINE CLZERO & LOCAL VARS
   (SETQ ENAME (ENTNEXT))      ;GET FIRST ENTITY
   (WHILE
      (NOT (NULL ENAME))       ;ARE THERE ANY MORE?
      (SETQ ELIST (ENTGET ENAME))
      (IF                      ;IF A LINE NOT ON LAYER 0
         (AND
           (EQUAL (CDR (ASSOC 0 ELIST )) "LINE")
           (NOT (EQUAL (CDR (ASSOC 8 ELIST)) "0"))
         )
         (PROGN                ;PUT IT ON LAYER 0
           (SETQ ELIST (SUBST (CONS 8 "0") (ASSOC 8 ELIST)
           (ENT MOD ELIST))
         )
      )
      (SETQ ENAME (ENTNEXT ENAME))  ;GET NEXT ENTIT
   )
)
```

1. Draw a CIRCLE, with a center point of 5,5 and a radius of 1.5, on layer o.
   Tjem type in the expressions.

EXPRESSION	VALUE
(setq ename (entlast))	<Entity name:>
(setq elist (entget ename))	((-1 . <Entity name:>) (0 . "CIRCLE") (8 . "0" (40 . 1.500000) (10 5.000000 5.000000))
(iqual (assoc o elist) "CIRCLE")	T
(<= (assoc 40 elist) 3.0)	T
(setq oldrad (assoc 40 elist))	(40 . 1.500000)
(setq newrad (cons 40 3.0))	(40 . 3.000000)
(setq e;ost (subst newrad oldrad elist))	((-1 . <Entit;y na;me:>) (0 . "CIRCLE") (8 . "0") (40 . 3.000000) (10 5.000000 5.000000))
(ENTMOD ELIST)	((-1 . <Entity name:>) (0 . "CIRCLE") (8 . "0") (40 . 3.000000) (10 5.000000 5.000000))

1. Modify the function CLZERO.LSP. Have it test for DIMENSION entities rather then LINE entities, and have it place each DIMENSION entity on Layer DIM Call the new function CDIMZERO.LSP.

2. Modify the function CDIMZERO.LSP. Have it EXPLODE every DIMENSION entity in the drawin. Place every new entity created by EXPLODEing the DIMENSION (line, text, etc.) on the same layer that the DIMENSION entity was originally placed on.
   Name the new function EXPLDINI.LSP.

רשימות מאפייני טבלאות .

AUTOCAD מחזיק מספר סוגי טבלאות.בינהן: LAYER,LTYPE,VIEW,STYLE,BLOCK.
כל אחת ניתנת לקריאה ישירות מה- ALISP.

ALISP כולל פקודות מיוחדות לטיפול בטבלאות אלו:

```
(SETQ LAYERTBL (TBLNEXT "LAYER" T))
```

הפקודה מכניסה לתוך **LAYERTBL** את הערך הראשון בטבלת ה- **LAYER** פעולת ה- **TBLNEXT** מזכירה את ה- **ENTNEXT**.

הערך המוחזר הוא בסגנון:

```
((0 . "LAYER") (2 . "0") (70 . 0) (62 . 7(6 . "DOTTED"))
```

ה-T בפקודת ה- **TBLNEXT** מורה להחזיר את הכניסה הראשונה בטבלה.
בלעדיו, הפקודה תביא את הכניסה הבאה.
ערכי הכניסה:

```
0  TABLE NAME
2  LAYER NAME
70 FROZEN
62 COLOR
6  LINETYPE
```

הפריט השני ברשימה, יכול להיות מוחזר בעזרת **CDR**

```
(CDR (ASSOC 6 (TBLNEXT "LAYER" T)    ---> "CONTINUOUS"
```

חיפוש בטבלה נעשה על פי שם הטבלה ושם הפריט המבוקש.

```
(SETQ STYLST (TBLSEARCH "STYLE" "STANDARD"))
```

<u>דוגמא:</u> פונקציה, הבודקת האם **LAYER** מסוים קיים בשרטוט:

```
;EXSTLYR.LSP
;
;FUNCTION PROMPTS FOR LAYER NAME
;IT DETERMINES WHETHER THE LAYER EXISTS IN THE CURRENT
;DRAWING AND RETURNS THE APPROPRIATE MESSAGE.
;
;WRITTEN BY PHIL CARTESIAN 10/15/87
;FOR AUTOCAD 2.6 AND LATER/
;
(DEFUN EXSTLYR (/ LNAME)
   (SETQ LNAME
      (GETSTRING "\nNAME OF LAYER: "))
   )
   (IF   (TBLSEARCH "LAYER" LNAME)
         (PROMPT (STRCAT "\nLAYER " LNAME "EXISTS!"))
         (PROMPT (STRCAT "\nLAYER " LNAME "NOT EXIST!"))
    )
    (PRIN1)
)
```

<u>תרגיל</u>

1.כתוב פונקציה, המבקשת מהמפעיל שם טבלה ושם פריט בטבלה , מחפשת את הטבלה ומודיעה אם הכניסה קיימת .

2.כתוב פונקציה, המבקשת שם LAYER חפש, אם השכבה הקיימת מציינת למפעיל את שם ה- LINETYPE ואת צבע השכבה ומחזירה SELECTION SET הכולל את כל הישויות באותה שכבה.

TABLE ASSOCIATION LISTS AND GROU;P CODES

LAYER

property	data type	group code	bit code
Table	string	0	
Name	string	2	
Frozen flag	Integer	70	1
Color	Integer	62	
(negative if the layer is turned off)			
Linetype	string	6	

LTYPE

property	data type	group code	bit code
Table	string	0	
Name	string	2	
Descriptive text	string	3	
Entity reference flag	Integer	70	64
Alignment code	Integer	72	
# of dash length items	Integer	73	
Total pattern length	Real	40	
Dash length 1	Real	49	
Dash length 2	Real	49	
(etc.)			

STYLE

property	data type	group code	bit code
Table	string	0	
Name	string	2	
Font file	string	3	
Bigfont file	string	4	
Fixed text height	Real	40	
Width factor	Real	41	
Last height used	Real	42	
Obliquing angle	Real	50	
Shape load request	Integer	70	1
Vertical style	Integer	70	4
Text backwards	Integer	71	2
Text upside down	Integer	71	4

VIEW

property	data type	group code	bit code
Table	string	0	
Name	string	2	
Center point	List	10	
Height	Real	40	
Width	Real	41	
Direction from origin	List	11	
Direction from origin	Real	31	
Not currently used	Integer	70	

BLOCK

property	data type	group code	bit code
Table	string	0	
Name	string	2	
Insertion base point	List	10	
Anonymous block	Integer	70	1
Has Attributes	Integer	70	2

קלט ופלט .

ALISP יכולה לקרוא ולכתוב לדיסק. קלט ופלט נעשה בעזרת פקודות **READ-LINE** ו- **WRITE-LINE**.

ובנוסף קימות שתי פעולות נוספות, הדרושות לתהליך **OPEN CLOSE**.

וראשית, ניצור קובץ בעזרת ה- **TEXT EDITOR**.בשם TEST.TXT.

```
FIRST LINE
SECOND LINE
THIRD LINE
```

פתיחת קובץ OPEN.

כדי לבצע כתיבה או קריאה, I/O יש תחילה לפתוח את הקובץ **OPEN**

OPEN מחזיר ערך הקרוי **FILE DESCRIPTOR** - שהוא כתובת שה- ALISP משתמש לגישה בשלב מאוחר יותר ב- **READ** או **WRITE**. כל אחת מפעולות אלה יש לבצע בעזרת מאפיין זה.

ב- OPEN מציינים את אשר עומדים לבצע -

קריאה	R-READ
כתיבה	W-WRITE
הוספה לקובץ	A-APPEND

```
<FILE #2D54> <---  (SETQ FILEPT (OPEN "TEST.TXT" "R"))
```

פתחנו את קובץ TEST.TXT לצורך קריאה ממנו.

```
(SETQ FILEPT (OPEN "TEST.TXT" "R"))
!FILEPT
(SETQ LIN1 (READ-LINE FILEPT)
!LIN1
(SETQ LIN2 (READ-LINE FILEPT)
(SETQ LIN3 (READ-LINE FILEPT))
(SETQ LIN4 (READ-LINE FILEPT))  ---> NIL
```

סגירת קובץ CLOSE:

```
NIL  <-- (SETQ FILEPT (CLOSE FILEPT))
```

כתיבה לקובץ : תחילה יש לפותחו לכתיבה -

```
(SETQ FILEPT (OPEN "TEST.TXT "W"))

(WRITE-LINE "NEW FIRST LINE" FILEPT)

(WRITE-LINE "NEW SECOND LINE" FILEPT)

(WRITE-LINE "NEW THIRD LINE" FILEPT)

(CLOSE FILEPT)
```

<u>תרגיל</u>

פתח קובץ חדש בשם TEXT.DOC, וכתוב את השורות הבאות לקובץ:

```
MY FIRST LINE
MY SECOND LINE
```

סגור הקובץ ופתח אותו ל- APPEND. והוסף השורה:

```
MY THIRD LINE
```

השתמש בפקודת TYPE להצגת הקובץ.

דוגמא: פונקצית IOTEXT, מוגדרת פקודת AUTOCAD חדשה הקוראת קובץ וכותבת אותו כ- TEXT בשרטוט AUTOCAD.

```
;IOTEXT.LSP
;
;COMMAND FUNCTION PROMPTS FOR NAME OF TEXT FILE.
;IT READ EACH LINE AND DRAWS THE CORRESPENDING TEXT IN ACAD
;IT ALSO PROMPTS FOR HEIGHT AND ROTATION ANGLE OF THE TEXT
;HOWEVER TEXT IS DRAWN LEFT-JUSTIFIED.
;
;WRITTEN BY PHIL CARTESIAN 2/6/88
;
;FOR AUTOCAD VERSION 2.5 AND LATER.
;
(DEFUN IODATA ()                                ;GET ALL DATA
  (SETQ
    FLNAM (GETSTRING "\nFILE NAME: ")           ;FILE NAME
    INSPT (GETPOINT  "\nINSERTION POINT ")      ;INS POINT
    TXTHT (GETDIST  INSPT  "\nHEIGHT :")        ;TEXT HEIGHT
    TXTRT (RTD (GETANGLE INSP \nROTATION: "))   ;TXT ROTATION
    INSHT (* TXTHT 1.5)                         ;INS POINT DIST
    INSRT (- (DTR TXTRT) (/ PI 2))              ;INS POINT ANGLE
    FLPNT (OPEN FLNAM "R")                      ;OPEN FILE
  )
)
;
(DEFUN DRTXT ()                                  ;DRAW TEXT
   (COMMAND "TEXT" INSPT TXTHT TXTRT TXTLN)
   (SETQ INSPT (POLAR INSPT INSRT INSHT))        ;NEXT INS POINT
   (SETQ TXTLN (READ-LINE FLNT))                 ;NEXT LINE
)
;
```

```
(DEFUN DTR (D)              ;CONVERT DEGREES TO RADIANS.
   (/ (* D PI) 180.0)
)
;
(DEFUN RTD (R)               ;CONVERT RADIAN TO DEGREES
   (/ (* R 180.0) PI)
)
;
(DEFUN C:IOTEXT ()           ;CONTROL ROUTINE
  (IODATA)                   ;GET DATA
  (COND
     ((EQUAL FLNT NIL)                     ;FILE EXISTS ?
      (PROMPT "\nFILE DOES NOT EXIST. ")
     )                                       ;FILE EMPTY ?
     ((EQUAL (SETQ TXTLN (READ-LINE FLPNT)) NIL)
      (PROMPT "\nFILE IS EMPTY. ")
     )
     (T
      (WHILE                               ;LOOP
        (NOT (NULL TXTLN))                 ;MORE TEXT  ???
        (DRWTXT)                           ;DRAW TEXT
      )
      (SETQ FLPNT (CLOSE FLPNT))           ;CLOSE FILE
     )
  )
)
```

ניהול משאבי זכרון ב- ALISP.

LISPHEAP-ל ו- **LISPSTACK** הינם שני משתנים בסביבת ה- **DOS** המציינים לAUTOCAD כמה זכרון עליו להקצות ל- ALISP.

HEAP הוא איזור הזכרון המשמש להגדרת פונקציות , STACK דרוש לשטח עבודה של ALISP שם היא מחזיקה את תוצאות הביניים במהלך פעילותיה.

הגדרות הגיוניות הינן:

SET LISPHEAP=40000

SET LISPSTACK=5000

הסברים נוספים ניתן למצוא ב- PERFORMANCE AND INSTALATION

בהודעת ALISP - **INSUFFICIENT NODE SPACE** כדאי להגדיל את ה- **HEAP**.

בהודעת ALISP - **OUT OF STACK SPACE** כדאי להגדיל את ה- **STACK**.

במערכת DOS רגילה, מקסימום השטח הדרוש ל-HEAP ול- STACK הוא 40000 .

ATOMLIST - הינה רשימת כל שמות ההגדרות והפונקציות ה- **ALISP** .
ניתן לראות אותה ב- ATOMLIST!

ניקוי ה- ATOMLIST - שטח ה- **LISPHEAP** מוגבל, ולכן קיים מספר מוגבל של פונקציות אותן ניתן להחזיק בזכרון. השיטה הבאה מאפשרת ניקוי ה- **ATOMLIST** כך שתשארנה רק הפונקציות הבנויות ב- **ALISP**.

```
( DEFUN CLEAN ()
      (SETQ ATOMLIST (MEMBER 'CLEAN ATOMLIST)
)
```

על מנת להחזיר את מצב ה- ATOMLIST, ובכך לשחרר מקום בזכרון, ניתן לבצע את השגרה **(CLEAN)**.

הפעלת **VMON**. - ניתן גם לטעון יותר פונקציות מהכמויות אותה מאפשר הזכרון על ידי איחסונן הזמני על הדיסק. הפעולה קרויה **PAGING** ומופעלת באמצעות פקודת **(VMON)**.

אזהרה: אין לבצע CLEAN אחרי שהפעלנו VMON. משום ש- VMON משתמש בATOMLIST בתור רשימת הצבעות לדיסק.!!!!!!

גודל פונקציה - פונקציות ארוכות,התופסות מקום רב ב- HEAP, מומלץ לשמור על אורך סביר לפונקציות.

אורך שמות משתנים - כללית, מומלץ לתת שמות משמעותיים לתכניות הנכתבות, על מנת שיהיה קל להבין אותן.
ניתן לשפר את ביצועי ה-ALISP אם המשתנים יקבלו שמות קצרים משש אותיות. זאת משום ש- **ALISP** שומר שם כזה ב- **NODE** אחד, בעוד ששמות ארוכים יותר הוא שומר בשנים.

פקודות AUTOLISP לפי קטגוריות

השמה:	SET ,INITGET ,SETVAR ,SETQ
AUTOCAD	GRDRAW ,GRREAD ,GRTEXT ,GRCLEAR ,GRAPHSCR ,COMMAND MENUCMD ,OSNAP ,REDRAW ,TBLNEXT ,TEXTSCR ,VER ,VPORTS
טיפול בישויות	ANGLE ,DISTANCE ,ENTDEL ,ENTGET ,ENTLAST ,ENTMOD ENTNEXT ,ENTSEL ,ENTUPD ,HANENT ,INTERS ,POLAR TBLNEXT ,TBLSEARCH ,TRANS
קבצים	OPEN ,READ-CHAR ,READ-LINE ,CLOSE ,FINDFILE ,LOAD WRITE-LINE ,WRITE-CHAR
קלט	GETPOINT ,GETANGLE ,GETCORNER ,GETDIST ,GETENV ,GETINT PAUSE ,GETORIENT ,GETREAL ,GETSTRING ,GETVAR ,GETKWORD PROMPT
LIST	CAR ,CDR .... CONS ,LAST ,LENGTH ,APPEND ,ATOMLIST NTH ,REVERSE ,SUBST ,LIST ,MEMBER
מתמטיקה	EXP ,COS ,ATAN ,BOOLE ,AND ,ABS 1- 1+ ~ / + - * LSH ,OR ,LOGIOR ,LOGAND ,LOG ,GCD ,FIX ,FLOAT ,EXPT SIN ,SQRT ,REM
זכרון	MEM ,GC
שונות	QUOTE ,EXPAND ,ALLOC ,*ERROR* ,VMON
פלט	PRIN1 ,PRINC ,PRINT ,TERPRI
פרוצדורה	MAPCAR ,LAMBDA ,IF ,FOREACH ,DEFUN ,EVAL ,APPLY WHILE ,PROGN ,REPEAT
עזרה בתכנות	UNTRACE ,TRACE
יחסים	MIN ,MAX ,COND ,ASSOC <= >= = > < /=
SELECTION SET	SSGET ,SSLENGTH ,SSMEMB ,SSNAME ,SSADD ,SSDEL
מחרוזות	READ ,STRCAT ,STRLEN ,SUBSTR
המרה ממחרוזת	RTOS ,ITOA ,CHR ,ATOI ,ATOF ,ASCII ,ANGTOS STRCASE
בדיקה	NULL ,NOT ,MINUSP ,LISTP ,EQUAL ,EQ ,BOUNDP ,ATOM NUMBERP ,TYPE ,ZEROP

תוכניות דוגמא:

```
(DEFUN C:BEYE   (/ CE CVP P1)          ;10 לגירסה BIRD'S EYE****
  (SETQ CE (GETVAR "CMDECHO")          ;שמירת ערך האקו
        CVP (GETVAR "CVPORT"))         ;CURRENT VIEWPORT
  (SETVAR "CMDECHO" 0)
  (INITGET 1)                          ;DISALLOW NULL INPUT
  (SETQ P1                             ;קבלת נקודה ראשונה
       (GETPOINT "\nSELECT WINDOW TO ZOOM IN ANY VPORT: "))
  (COMMAND "VIEW" "WINDOW" "$ZWAV" P1 PAUSE)  ;נקודה שניה
  (SETVAR "CVPORT" CVP)                ;מחזיר אותנו חזרה חלון
  (COMMAND "VIEW" "RESTORE" "$ZWAV")   ;טוען את החלון שהגדרנו
  (COMMAND "VIEW" "DELETE"  "$ZWAV")   ;מבטל את ההגדרה
  (SETVAR "CMDECHO" CE)
  (PRIN1)
)
```

קביעת **ELEVATION** לפי ישות נבחרת:

תחילה דוגמא ל- **ENTGET** ול- **ENTSEL**

```
COMMAND:LINE 1,1 6,6 RETURN
(SETQ E (ENTSEL "PLEASE CHOOSE ENTITY:"))
3,3
(<ENTITY NAME: 6000014> (3.0 3.0 0.0))
(SETQ A (ENTGET (ENTLAST)))
                                       ENTITY        LAYER
( (-1 <ENTITY NAME: 60000014>)   (0 . "LINE") (8 . "0")
(38 . 25.0) (39 . 1.5) (10 1.0 2.0 0.0) (11 6.0 6.0 0.0) )
ELEV            THICK          START             END
```

```
(DEFUN C:ELSET ( / E T)
(PRINC (STRCAT "\nSELECT AN ENTITY TO MAKE ITS ELEVATION "
               "AND THICKNESS THE CURRENT"))
 (SETQ
    E (ENTGET (CAR (ENTSEL "SETTING:"))) ;מקבל את שם ישות
    T (CDR (ASSOC 39 E))                 ;עובי
  )
  (IF (NOT T) (SETQ T 0.0))
  (SETQ E (CDR (ASSOC 38 E)))            ;ELEVATION
  (IF (NOT E) (SETQ E 0.0))
  (COMMAND "ELEV" E T )
  (PRINC (STRCAT "\nCURRENT ELEVATION=" (RTOS E)
                 "\nCURRENT THICKNESS=" (RTOS T)))     )
```

דוגמא להצבת נקודה על קו, במרחק מאחד הקצוות:

```
(DEFUN GET (A B) (CDR (ASSOC A B)))

(DEFUN C:ALONG (/ ENAME EDATA PT0 PT1 PT2 DIST ANG)
  (SETVAR "CMDECHO" 0)
  (SETQ ENAME (ENTSEL "\nSELECT LINE : ")
        EDATA (ENTGET (CAR ENAME))
        PT0   (OSNAP (CADR ENAME) "END")
        PT1   (GET 10 EDATA)
        PT2   (GET 11 EDATA)
        DIST  (GETDIST "\nENTER DISTANCE:")
        PT1   (IF (< (DISTANCE PT0 PT1) 1.0E-6) PT2 PT1)
        ANG   (ANGLE PT0 PT1)
  )
  (COMMAND ".POINT" (POLAR PT0 ANG DIST))
)
```

הפעלה ע"י **ALONG**, כאשר כמובן כבר קיים קו.

כתיבת הערות החוצה, בזמן העבודה, לקובץ בשם שם שרטוט.NOT:

```
(DEFUN C:NOTES ()
  (COMMAND "EDIT" (STRCAT (GETVAR "DWGNAME") ".NOT"))
)
```

```
;                              RELATIVE.LSP

;   By Simon Jones  Autodesk Ltd, London     23 July 1986

;     This macro provides a point that is relative by a
;     given angle and distance to a known point such as
;     the "End pt" of a line or "Centre of" a circle.

;     When AutoCAD requires a point for a command
;     respond with:
;                  !(relative)   (from keyboard)

;     Or alternatively enter the following line as a menu
;    command in the "**OSNAPB 3" sub-menu in the screen menu
;    with the remaining object snap commands:

;            [RELATIVE]!(offset) $s=

;    The macro can be copied onto an ACAD.LSP file to ensure
;    it is allows loaded into a drawing

;              then:
;                   1) Enter or select a point.
;                       (using object snap if required)
;                   2) Enter or show a distance relative to
;                       the selected point.
;                   3) Enter or show an angle relative to
;                       the selected point

      (defun RELATIVE ()
         (setq pts (getpoint "\nRelative to: "))
         (setq dist (getdist pts "\nRelative distance: "))
         (setq ang (getangle pts "\nRelative angle: "))
         (setq pt (polar pts ang dist))
      )


;                              REF.LSP

; This is a useful routine for obtaining a relative point.
;Itcan be used whenever an AutoCAD command requests a point.
;Just enter"(ref)"in response to the "...point:" prompt, and
;enter the desired base(reference)pointandthe relative/polar
; offset from that point. For example:

;Command: LINE
;From point: (ref)
;Reference point: (...pick a point...)
;Enter relative/polar coordinate(with@): @x,y or @dist<angle
;  To point:

(defun ref ()
  (setvar "LASTPOINT" (getpoint "Reference point: "))
  (getpoint "\nEnter relative/polar coordinates (with @): ")        )
```

הדפסת קובץ על המסך בישיבת AUTOCAD

```
;
;Function to print (list) an ASCII text file on the screen.
;
;  Usage:   (fprint "filename.ext")
;
(defun fprint (s / c i)
   (setq i (open s "r"))
   (if i
      (progn
         (while (setq c (read-line i))
            (princ c)
            (princ "\n")
         )
         (close i)
      )
      (princ (strcat "Cannot open file " s))
   )
   (princ)
)
```

הצגת שם השרטוט, בתוך מסגרת ה- ACAD:

```
(GRTEXT 24 "DWG #")
(GRTEXT 25 (GETVAR ("DWGNAME"))
```

הצבת נקודה, באמצע המרחק בין שתי נקודות (גם ב- 3D):

```
(DEFUN BETWEEN (/ A B  PT)
   (SETQ A (GETPOINT  "\nPOINT 1 ? "))
   (SETQ B (GETPOINT  "\nPOINT 2 ? "))
   (SETQ PT (LIST  (/ (+ (CAR A) (CAR B))  2.0)
                   (/ (+ (CADR A) (CADR B)) 2.0)))
   (IF (CADDR A)
     (SETQ PT (APPEND PT
                (LIST (/ (+ (CADDR A) (CADDR B)) 2.0))
              )
     )
   )
 )
```

הפעלה ע"י:

```
COMMAND:LINE
FROM POINT : (BETWEEN)......
```

דרך נוחה להדפיס ערך משתנים, בזמן ניקוי תוכנית מ-BUGS:

```
(DEFUN PRN (LST / N)
     (FOREACH N LST (PROGN (PRINT N)
          (PRIN1 (EVAL N))))
     (PRIN1)
)
```

הפעלה ע"י נטיעת שורה מסוג בתוכנית הנבדקת (PRN '(VAR1 VAR2:

סיבוב שרטוט שלם ב-90 מעלות:

```
(DEFUN C:FLIP (/ )
 (SETVAR "CMDECHO" 0)
 (SETVAR "HIGHLIGHT" 0)
 (SETQ   MIN (GETVAR "LIMMIN")
         MAX (GETVAR "LIMMAX")
         HAF (/ (CAR MAX) 2)
         PT1 (LIST HAF HAF)
         MX (LIST (CADR MAX) (CAR MAX))
  )
  (SETVAR "LIMMAX" MX)
  (COMMAND "ROTATE" "C" MIN MAX " " PT1 "-90" "ZOOM" "A"
           "VIEW" "S" "O")
)
```

תאריך השאלה	תאריך השאלה
ספריה 11 -[illegible]- 1994	ספריה 0[illegible] -12- 1997
ספריה 26 -01- 1995	ספריה 16 -[illegible]- 1997
[illegible]ריה 19 -[illegible]- 1995	ספריה [illegible] -02- 1998
22 -0[illegible]- 1995	[illegible]ריה [illegible] -08- 1998
[illegible]יה [illegible] 1997	